Theoretical Issues and Practical Considerations Concerning Confidence Measures for Multi-Layer Perceptrons

Georgios Papadopoulos

A thesis submitted for the degree of Doctor of Philosophy.
The University of Edinburgh.
November 2000

Abstract

Over the last fifteen years multi-layer perceptrons (MLPs) have been applied to a wide range of non-linear modelling problems in industrial, medical or financial applications. System reliability and usability are increased if predictions are supported by an associated confidence measure (CM). Several approaches to prediction confidence estimation have been reported. However, the problem of applying and assessing the performance of these techniques in real applications has not been the subject of detailed study.

Therefore, the primary aim of this thesis is to study existing CM methods and assess their practicability and performance in harsh real-world environments. The motivation for this work was a real industrial application - the development of a paper curl prediction system. Curl is an important paper quality parameter that can only be measured after production. The available data were sparse and were known to be corrupted by possibly gross errors. Furthermore, it was suspected that data noise was not constant over input space.

Three approaches were identified as suitable for use in real-world applications: maximum likelihood (ML), the approximate Bayesian approach and the bootstrap technique. These methods were initially compared using a standard CM performance evaluation method, based on estimating the prediction interval coverage probability (PI CP). It was found that the PI CP metric can only gauge CM performance as an average over the input space. However, local CM performance is crucial because a CM must associate low confidence with high data noise/low data density regions and high confidence with low noise/high data density regions. Moreover, evaluating local performance could be used to gauge the input-dependency of the noise in the data. For this reason, a new CM evaluation technique was developed to study local CM performance. The new approach, called classification of local uncertainty estimates (CLUES), was then used for a new comparison study, this time in the light of local performance. Three main conclusions were reached: the noise in the curl data was found to have input-dependent variance, the approximate Bayesian approach outperformed the other two in most cases, and the bootstrap technique was found to be inferior to both ML and Bayesian methods for data sets of input-dependent data noise variance.

In principle, the techniques used in this thesis may also be applied to classification tasks as well as other real-world modelling applications with sparse, noisy data sets.

Declaration of originality

I hereby declare that, except where otherwise stated, the research recorded in this thesis and the thesis itself was composed and originated entirely by myself in the Department of Electronics and Electrical Engineering at The University of Edinburgh.

Georgios Papadopoulos

Acknowledgements

This thesis was made possible through the support of EPSRC (grant number GR/L30930) and Tullis Russel & Co. Ltd. Moreover, I would like to thank my supervisor Prof. Alan F. Murray and Dr. Peter J. Edwards for their help and support during the course of this work. I would also like to thank all my friends and especially Manolis, Vaggelis, Giorgos, and Jan.

Finally, I would like to thank my parents without whom this thesis would not have been completed.

Contents

List of figures

List of tables

Acronyms and abbreviations

ANN	Artificial Neural Network
CI	Confidence Interval
CLUES	Classification of Local Uncertainty EstimateS
CM	Confidence Measure
CP	Coverage Probability
EF	Evidence Framework
ML	Maximum Likelihood
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NN	Neural Network
PI	Prediction Interval
PCA	Principal Components Analysis
PDF	Probability Density Function
SVD	Singular Value Decomposition
TR	Tullis-Russell

Nomenclature

The mathematical notation in this thesis follows the following general principles:

- Vectors are denoted by lower-case bold letters, *e.g.* $\mathbf{x}$.
- Matrices are denoted by upper-case bold letters, *e.g.* $\mathbf{H}$.
- Sets are denoted by calligraphical letters, *e.g.* $\mathcal{C}_i$.
- A hat on a letter indicates estimated values, *e.g.* $\hat{\mathbf{w}}$ denotes the vector of estimated weight values.
- A dot on a letter indicates the optimum or true value of a parameter, *e.g.* $\dot{\mathbf{w}}$ is the vector of optimum weight values.

Some of the most commonly used symbols and a short description of them follow:

a_w	regularization parameter for regression weights
a_u	regularization parameter for variance weights
C	number of confidence classes
$\mathcal{C}_i$	confidence class set
$C(\mathbf{w})$	mean squared error including the weight decay term
$\mathcal{D}$	training data set
$\mathcal{D}_i^*$	bootstrap replicate set
$\mathcal{D} - \mathcal{D}_i^*$	bootstrap out-of-sample set
e	additive noise component
$E(\mathbf{w})$	mean squared error function
$f(\mathbf{x})$	smooth function to be modelled by the network
$\mathbf{H}$	exact Hessian matrix
$\tilde{\mathbf{H}}$	approximate Hessian matrix
$\mathbf{I}$	unitary matrix
$\mathbf{J}$	Jacobian matrix

K	number of hidden units
M	dimensionality of input vector
N	number of training patterns
N_t	number of test patterns
$N(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2
$p(\mathbf{x})$	probability density of pattern $\mathbf{x}$
$r(\mathbf{x})$	absolute prediction error
r_i	average $r(\mathbf{x})$ for confidence class $\mathcal{C}_i$
s_i	standard deviation threshold for confidence class $\mathcal{C}_i$
$\mathcal{S}_i$	set of patterns classified as belonging to class $\mathcal{C}_i$
t	target
$\mathbf{u}$	variance weight vector
U	number of variance weights and biases
$\mathbf{w}$	regression weight vector
W	number of regression weights and biases
$\mathbf{x}$	input vector
$y(\mathbf{x})$ $y(\mathbf{x}; \mathbf{w})$	network output, *i.e.* regression estimate
σ_ν^2	data noise variance
$\hat{\sigma}_\nu^2(\mathbf{x}; \mathbf{u})$	network estimate of data noise variance
σ_m^2	model uncertainty variance
$\hat{\sigma}_{\text{cp}}$	standard deviation of the coverage probability

Chapter 1
Introduction and thesis overview

1.1 Artificial neural networks and prediction confidence

Artificial neural networks (ANNs) are non-linear statistical tools which can be used for solving complex problems in fields such as medical condition monitoring [1–3], forecasting [4–8], or prediction and process control [9–14]. An important issue for every automatic prediction or control system is the estimation of the *reliability* or *confidence* associated with the system output. Accompanying a network output with a confidence estimate can significantly increase system reliability and usability in real-world applications.

1.2 Solving real problems with ANNs

There exist two basic types of neural network learning:

- *Supervised* learning where the desirable answer (target) is known.
- *Unsupervised* learning where there are no clearly defined targets and the goal may instead be the modelling of the data probability density or the discovery of clusters in the data.

This thesis examines the issue of confidence estimation for *multi-layer perceptrons* (MLPs), a particular category of supervised neural networks. MLPs are non-linear, statistical models widely used for problem solving in a variety of real applications. The main advantage of MLPs as modelling tools is their ability to infer the input-output mapping directly from the data, *i.e.* no additional knowledge about the form of the mapping is necessary.

Supervised learning problems can be further divided into *regression* and *classification* tasks. A regression task involves the estimation of some function from a set of possibly noisy, input-target examples. In classification tasks the goal is to classify input patterns as belonging to a number of predefined classes by determining the appropriate decision boundaries. In this thesis, regression tasks are discussed but all the methods can be adapted for classification tasks as well.

1.3 Prediction confidence

A conventional MLP employed in a regression task would typically output a single prediction when presented with a previously unseen input pattern. In real situations, the user of such a system must then decide (based on his/her experience) if the network prediction should be trusted or not. The usability of such systems is therefore limited because spurious predictions cannot be distinguished from predictions based on the information contained in the training data. It is therefore highly desirable to provide an automatic confidence estimate associated with the network prediction. This can be done by incorporating a *confidence measure* (CM) component in the system design. A confidence measure is an estimate of the reliability associated with the network prediction.

Confidence estimation for regression tasks typically involves the estimation of some *error bounds* or *intervals* around the network prediction [15–17]. Alternatively, a *confidence index* (*e.g.* valid-invalid prediction) can be used to discard input patterns that originate from input space regions not represented in the training data [18].

This thesis mainly examines the formulation of reliable *prediction intervals* [17, 19, 20] around the network solution. The issue of identifying inputs originating from regions not represented in the training data, *i.e. novel inputs*, is also briefly examined. Prediction intervals are error bars within which the true target is known to lie with a given probability. They are sometimes confused with *confidence* intervals (see fig. 1.1) within which the true regression is known to lie with a given probability. In this thesis prediction intervals are mainly considered because a real prediction system effectively predicts the targets (*i.e.* noisy values) rather than the noise-free regression function. These concepts are defined and discussed in detail in chapters 2 and 4.

1.4 The original project goals

The motivation for this work comes from the development of a neural network system for the prediction of paper *curl* [21, 22]. Curl, a tendency of the paper to be non-flat, is an important paper quality parameter that can only be measured after production. High curl results in customer dissatisfaction and waste of plant and engineering time [22]. Curl prediction is a multi-dimensional, non-linear regression problem [21]. The existing prediction system has been successfully tested in the *Tullis-Russell* paper-making plant, the company that provided

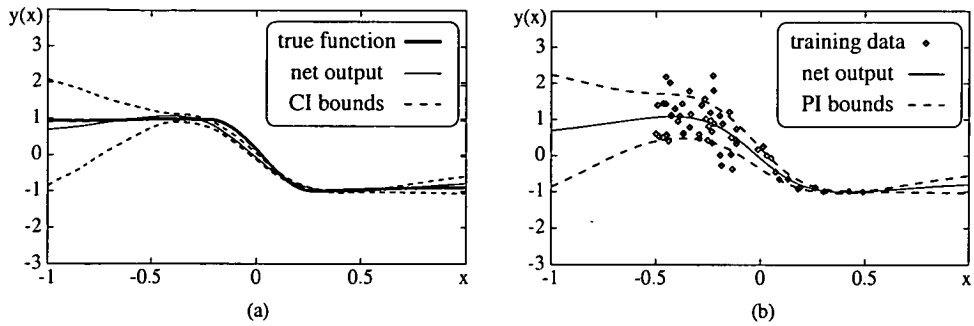


Figure 1.1: *Confidence and prediction intervals. Graph (a) shows the confidence intervals (CIs) for an one-dimensional regression task solved with a neural network. Graph (b) shows the prediction intervals (PIs) for the same task and network. Note that the prediction intervals are wider since they include uncertainty due to the noise in the data as well as due to the inaccuracies of the neural network. Confidence intervals, on the other hand, are only concerned with uncertainty due to the network. These concepts will be discussed in detail in chapters 2 and 4.*

the data set. The curl prediction system uses 10 parameters of the paper manufacture process as inputs and predicts the value of curl before production is completed. However, the system employs a simplistic approach to confidence estimation which is unlikely to give accurate results.

Although there exist many confidence estimation techniques for MLPs, the issue of assessing and comparing these techniques under real circumstances has received little systematic study. Therefore, the original goal of this project was to investigate existing methods for confidence estimation in neural networks and identify and compare methods appropriate for use in the curl estimator. It was expected that this would help determine the most appropriate confidence estimation approach to be incorporated in the curl estimation system. Moreover, this study would be of a more general interest because its conclusions should be applicable to other applications similar to the curl task.

It must be stressed that the aim of this comparison study is not to determine the theoretically best confidence estimation technique. Instead the focus is on methods that can be pragmatically applied to the curl task and possibly other similar industrial applications. Such methods must take into account the practical limitations imposed on any industrial system. Apart from the limited available budget and computer power, the final system must be easily retrained by non-experts. This is necessary since conditions in the paper-making plant may change with time rendering the system obsolete. Moreover, the training time must be reasonably short (although this requirement can be relaxed since training can be performed off-line) and the "query" time

(*i.e.* the time needed for the system to produce an answer once presented with an input pattern) must be reasonably small.

These practical issues may seriously affect the usability and thus acceptance of an automatic control system by the industry. Therefore, the ideal method should not be too computationally demanding. Naturally, as technology progresses and more powerful computers become available at a lower price, more advanced techniques can be used to solve real-world problems. The conclusions and choices in this thesis take into account the situation in the market as it was during the course of this work.

1.5 Project evolution

As the initial literature survey progressed, it became clear that the choice of a confidence estimation approach conditions, or is equivalent to, a choice of network training and regression estimation approach. Therefore, in practice, the two issues, regression and confidence estimation, should be viewed in tandem. However, this thesis focuses on the issue of confidence estimation, *i.e.* an extensive study of the generalization capabilities of each method is not included.

In the first stage of this work, the methods appropriate for use in a real neural network system were identified and a comparison study of these techniques was performed. This first comparison study used the *prediction interval coverage probability* (PI CP) [15, 16, 20], a commonly used approach to assessing CM performance. An important modelling decision is whether to model the variance of the noise inherent in the data as constant or dependent on the inputs. Comparing both approaches on artificial data it was found that the PI CP metric is not sensitive to the adopted assumption. In other words, no improvement in terms of PI CP occurs when the adopted assumption about data noise variance matches the actual input-dependency of data noise. This is because the PI CP can only assess the quality of the confidence estimate as an *average* over the entire input space, rather than *locally*.

A confidence estimation technique should yield accurate confidence estimates in all input space regions, in other words, locally good CM performance is crucial. Therefore, it became apparent that the PI CP is inadequate and a new CM performance evaluation technique should be developed. The new technique should be sensitive to *local* rather than *global* CM performance.

After these issues were identified, the rest of this work was conducted with the following aims:

- Development of a confidence measure evaluation method that quantifies the local quality of the confidence estimate.
- Repeat of the comparison study using the new method.

No other region-dependent approach to CM performance evaluation has been reported in the literature. The method, once developed, was used to investigate the following issues:

- The input-dependency of the noise in the curl data.
- The effect of including or not the data noise variance estimate as a weighting term in the learning rate of the regression network weights.

The over-arching aim of this study was, therefore, to examine the relationship between the characteristics of data, the neural training approach and the optimal confidence estimation technique for practical applications.

1.6 Thesis structure and overview

At a first level, this work is a comparison study of three commonly used approaches to confidence estimation. The performance of these techniques is assessed using artificial as well as the real curl data. The comparison study is performed at first using the PI CP metric, a conventional confidence measure evaluation technique. The failure of this method to assess local performance is demonstrated and the need for a new approach justified. Subsequently, a new comparison approach is developed and the confidence estimation techniques are reevaluated and compared in the light of local performance. Therefore, on a different level, this work can be viewed as a criticism of the PI CP metric which led to the formulation of an alternative, region-dependent, CM evaluation metric.

The rest of this thesis is therefore structured as follows:

- Chapter 2 presents a review of the problem of confidence estimation for MLPs and describes available confidence estimation techniques for neural networks.

- Chapter 3 investigates the issue of detecting novel inputs. Such inputs should be assigned very low confidence or even be discarded because they originate from input space regions which are not represented in the training data. The ability of estimates of model uncertainty variance to identify novel inputs is investigated and compared to that of a data density measure.

- Chapter 4 reviews existing methods for evaluating confidence estimation performance. The PI CP metric, a commonly used method, is then employed for a first comparison of the considered techniques using both artificial and real data sets. A novel comparison study is presented treating data noise variance as a function of the inputs.

- Chapter 5 describes the new approach to confidence estimation evaluation. This method measures performance locally rather than globally over the entire input space. Results of comparative studies using this approach are presented for artificial and real data.

- Finally, chapter 6 provides a summary of the thesis and states the final conclusions and possible future work in the field.

Chapter 2
Background to confidence estimation for MLPs

2.1 Introduction

As mentioned in chapter 1, this work focuses on regression tasks since the curl estimation task was originally defined as a regression task. All presented methods can be adapted for classification tasks as well, however, the details are beyond the scope of this thesis.

Existing approaches to confidence estimation for neural networks can be divided into methods originating from non-linear regression theory [15, 23–25], maximum likelihood theory [26–28], Bayesian statistics [17, 29–32] and bootstrap methods [2, 19, 33]. All these methods compute prediction uncertainty by estimating the variance of the target or output probability distribution. Therefore, it is necessary to assume that the target and output distributions take some convenient form (*e.g.* Gaussian).

There exists a different approach to confidence estimation which employs the *probability density function* (PDF) of the training data to identify input patterns that significantly differ from the training data. This process is called *novelty detection* [18] and is discussed in chapter three as it is unrelated to the other methods but necessary for the final confidence estimation system.

This chapter

- introduces some basic terminology, symbolism and concepts used in the rest of the thesis,
- introduces the basic assumptions adopted in this study, necessary for the confidence estimation methods,
- introduces the considered confidence estimation techniques and justifies why other methods were not considered as candidates.

The chapter is organised as follows:

Section 2.2 introduces the regression estimation problem using MLPs.

Section 2.3 discusses the sources of uncertainty contributing in the total uncertainty associated with MLP predictions.

Section 2.4 introduces the considered approaches to neural network training and confidence estimation.

Section 2.5 discusses committees of networks and their effect on uncertainty.

2.2 Regression using MLPs

This section introduces some basic terminology and issues concerning MLPs and regression problems. Some of these issues, like network training and model architecture selection, are major subjects of ongoing research. Such issues are only briefly introduced in order to put the approach used in this thesis in context with other alternative methods.

2.2.1 Training data

In a non-linear regression task the available data set $\mathcal{D}$ consists of N examples of input-target pairs $(\mathbf{x}^n, t^n)$, $n = 1, ..., N$. For simplicity, one-dimensional targets are considered in this thesis. When the target is multi-dimensional all methods apply directly to each output separately if there is no *covariance* between the outputs[1]. However, in the curl problem as well as other similar applications, the output is one-dimensional. Therefore, considering one-dimensional targets suffices for the aims of this work. The target is assumed to be connected with the input through a non-linear, smooth function $f(\mathbf{x})$ plus some additive noise e. Thus, target t^n is given as

$$t^n = f(\mathbf{x}^n) + e^n \tag{2.1}$$

The error component e^n is due to noise inherent to the data. For example, it can be due to errors in the measurement of the output or human error during data collection.

Typically, training data are *pre-processed* before used to train a neural network. Likewise,

[1] For a treatment of the multi-variate output case using the full variance-covariance matrix see [28].

network outputs may be *post-processed* (to remove the effect of pre-processing) before presented to the user. Data pre-processing significantly simplifies network training. It may involve normalization, encoding of categorical variables, feature extraction [34], principal components analysis (PCA) [35] and dimensionality reduction.

2.2.2 Network architecture

A neural network model based on the MLP architecture can be used to estimate function $f(\mathbf{x})$. In general a number of different architectures should be tried before the optimum is selected. This process is known as *model selection* and is a major subject of research. A number of algorithms have been proposed to simplify model selection. These include network pruning [36–38] or growing algorithms [39, 40], as well as Bayesian model comparison methods based on evaluation of the *evidence* for the model [34, 41].

For simplicity, in this thesis, the architecture is restricted to two-layer networks (see fig. 2.1) with units of *logistic sigmoid* activation function. The logistic sigmoid function is given by [34]

$$g(x) \equiv \frac{1}{1+\exp(-x)} \tag{2.2}$$

Therefore, the activation (output) of hidden unit j is given by

$$y_j(\mathbf{x}) = g(\sum_{i=1}^{M} w_{ji} x_i) \tag{2.3}$$

where M is the number of input parameters, w_{ji} is the weight of the connection between input i and hidden unit j, and $\mathbf{x} = \{x_1, ..., x_M\}$ is the input vector. Biases have been included in the sum by introducing units with activation fixed at $+1$. In a regression network, the output unit usually has a linear activation function. Therefore, the output of a two-layer network with hidden units of sigmoidal activation and linear output unit is given by

$$y(\mathbf{x}; \mathbf{w}) = \sum_{j=1}^{K} w_j y_j(\mathbf{x}) \tag{2.4}$$

where K is the number of hidden units and w_j is the weight of the connection between hidden unit j and the output unit. The total number of weights W (counting biases as weights) in this

network is

$$W = KM + K \tag{2.5}$$

It has been shown that a two-layer MLP with sigmoidal hidden units is a *universal approximator* [42, 43]. That is, it can approximate arbitrarily well any continuous functional mapping from a finite dimensional space to another. Thus, by considering two-layer architectures no loss of generality occurs. Moreover, in this case, model selection reduces to choosing the optimum number of units in the hidden layer.

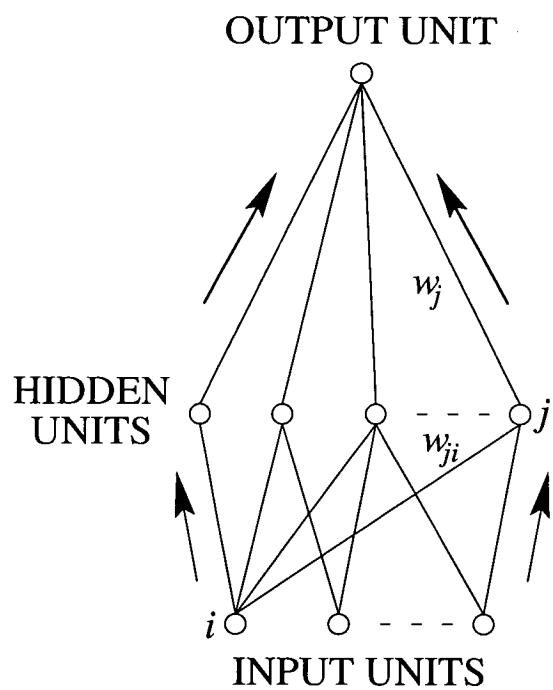


Figure 2.1: *The two-layer MLP architecture. For regression tasks the hidden units have sigmoid activation while the output unit linear activation function. In general the input and hidden layers are fully connected. Only part of the connections are shown in the graph for clarity.*

2.2.3 Network training

A neural network trained on data set $\mathcal{D}$ learns to output $y(\mathbf{x}; \hat{\mathbf{w}})$ the network estimate of the true regression function $f(\mathbf{x})$. Vector $\hat{\mathbf{w}}$ contains the estimated weight values. Effectively the network *generalizes* over the noise in the data to obtain an estimate of the true input-target relationship. Neural networks are trained using some non-linear optimisation technique, such

as *gradient descent* (also known as *steepest descent*) [34]. Initially the weights are assigned some random values and the algorithm seeks to minimise the *error function* with respect to the weights. The most common error function for regression learning is the sum-of-squares error or mean squared error (MSE)

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}[t^n - y(\mathbf{x}^n;\mathbf{w})]^2 \tag{2.6}$$

Choosing the parameters by minimising this error function corresponds to minimising the squares of the vertical deviations of the network function from the targets $[t^n - y(\mathbf{x}^n;\mathbf{w})]^2$, $n = 1,...,N$. In statistical theory, this approach to parameter estimation is known as *least squares* [44].

The training algorithm uses the derivatives of the error function with respect to the weights to compute the steepest direction and then takes a step towards this direction by appropriately perturbing the weight values. The derivatives of the error with respect to the weights can be calculated using the error *back-propagation* algorithm [45]. In practice this procedure is susceptible to the choice of initial weights and must be repeated several times until a satisfactory solution is reached. Moreover, the size of the step (towards the optimum direction), called *learning rate* must be set beforehand. There exist various modifications of the algorithm that seek to alleviate these disadvantages. These include the addition of a *momentum* term [46] or the *bold driver* technique [47,48] which seeks to set the learning rate and momentum parameters automatically. More principled alternatives include *line search* [49,50], *conjugate gradients* [51,52], *quasi-Newton* or *variable metric* methods [48,53,54] and the *Levenberg-Marquardt* algorithm [55,56]. In most of these methods learning is viewed as a two-stage procedure. First, the optimum direction of error decrease is determined and then the optimum step size is chosen.

In the simplest case, the optimum direction can be the steepest direction, found using gradient information and back-propagation. Alternatively, more complex techniques, such as conjugate gradients or variable metric, can be used. The simplest way of determining the optimum step along the determined direction of move is to try several steps of varying sizes and stop when the error starts to increase [57]. More efficient line search methods like the Brent algorithm [50,58] provide a better alternative.

In general, the error function of a neural network solving a complex problem may have more than one minimum - not counting multiple minima due to network symmetries. Thus, the

training procedure may actually reach a local minimum instead of a global one. Moreover, even if the global minimum is found, the obtained weight values $\hat{\mathbf{w}}$ are still estimates of the optimum values $\dot{\mathbf{w}}$ due to the fact that a finite and typically small data set is used. Therefore, the network output is only an estimate of the true regression due to the incomplete data set (not all possible input-output pairs are available) and the inefficiencies of the network architecture selection and training procedures. Even assuming that the network architecture is a good fit to the problem the training algorithm still introduces uncertainty.

In practice, training is usually stopped before a minimum on the training set is reached to avoid *overfitting*, *i.e.* learning the noise rather than the smooth function $f(\mathbf{x})$. Various techniques such as using a *validation* set for *early-stopping* or introducing a *regularization* term in the error can be used to avoid overfitting.

A validation set is a set of data which has not been used for training the network. This set can be used to compute an unbiased error estimate (*i.e.* different from the training error), called validation error. Training can be stopped when the validation error starts to increase. The drawback of this approach is that the available data must be divided in three subsets, which are used for training, validating and testing the final network respectively. As these three subsets must be of similar size, a high proportion of the available data is not used for training. This problem can be alleviated using *cross-validation* [59–61]. In cross-validation the data set is divided in S subsets and training is repeated S times, each time using one subset for validation and the remaining data for training. The error is then averaged over the S tries[2]. Thus, a high proportion of the data $(1 - 1/S)$ is still used for training. However, this procedure requires training to be repeated S times.

A regularization term can be added to the error function to encourage smoother network mappings and thus prevent the occurrence of overfitting. The most common form of regularization is *weight decay* [46]. Weight decay is based on the observation that weights must be assigned small values for the network mapping to be smooth. Small weight values can be encouraged by including in the error function a penalty term proportional to the square of the weight values. For example, if the mean-squared error function is used, the total error including the regulariz-

[2]Note that cross-validation may also be used as an alternative committee formation technique. See section 2.5.

ation term is given by

$$C(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}[t^n - y(\mathbf{x}^n;\mathbf{w})]^2 + \frac{\alpha_w}{2}\sum_{i=1}^{W} w_i^2 \tag{2.7}$$

where it is assumed that the value of the regularization parameter α_w is the same for all weights w_i. In general, each weight or family of weights can be assigned a different regularization parameter. It has been shown that using weight decay can lead to significant improvement in generalization performance [62].

2.3 Sources of uncertainty in MLP predictions

Neural network predictions suffer uncertainty due to inaccuracies of the training data and to the limitations of the model. As mentioned in sections 2.2.1 and 2.2.3 the training set is typically noisy - *i.e.* the available target values are corrupted by errors - and incomplete (not all possible input-output examples are available). The iterative optimisation algorithm used to train the network gives rise to a second source of uncertainty, which is manifested as model misspecification (*e.g.* weight value misspecification).

This section presents the sources of uncertainty associated with MLP predictions in more detail. The assumptions used in the rest of this thesis are also presented. Finally, a discussion of the validity of these assumptions with respect to the curl data is given.

2.3.1 Data Noise

All real data sets contain measurement errors called data noise. In general, both inputs and target can be noisy. Regression using noisy inputs [63, 64] is a difficult task beyond the scope of this work. In this thesis the inputs are assumed correct and only noise associated with the targets is considered (see eq. (2.1)). Section 2.3.4 explains why this assumption is not expected to significantly degrade confidence estimation for the curl task.

Typically, in non-linear regression problems such as the curl problem the additive errors are assumed to be independently and identically distributed [15, 16, 23]. This means that each error e^n, $n = 1, ..., N$ is independent of each other and they all have the same distribution. The most commonly used distribution is the *normal* or *Gaussian* distribution with zero mean and

variance σ_ν^2 which may be constant or dependent on the inputs. For small data sets, a Student's t distribution may give more accurate error representation [33, 65]. In the rest of this thesis the variance of the noise in the data is referred to as *data noise variance* or simply noise variance.

Data noise variance is commonly taken to be constant over input space. However, in complex neural networks applications this assumption will often be incorrect. According to [66], cases of data of non-constant variance (*heteroscedasticity*, in statistical parlance) may arise due to

1. the use of averaged data,
2. variances depending on the means or input variables,
3. different sources of data *e.g.* different observers, sites, *etc* or
4. the presence of *outliers*, *i.e.* a small number of data corrupted by very large errors.

In such cases, data noise variance may be seriously misjudged if the erroneous, constant variance assumption is used. As far as the curl data are concerned, it is suspected that conditions 2 and 3 may apply. Condition 2 may arise due to the method used to measure curl, while condition 3 may apply because the curl data set contains patterns of different paper grade. In this thesis, both noise variance assumptions are considered for the curl data.

2.3.2 Uncertainty associated with the model

Data noise is not the only uncertainty source. The neural network algorithm itself introduces uncertainty due to model misspecification and inefficiencies of the training method[3]. A network trained on a finite data set learns a better representation of the data in regions of high input data density as it can only learn to represent information contained in the training set [18]. Moreover, because of the nature of the training algorithm, there is no guarantee that the weight values correspond to the global minimum of the error function. Even if the global minimum is found the solution will not necessarily be the optimum because the finite training set only partly describes the true data generating mechanism. In the rest of the thesis, the uncertainty due to the model is called *model uncertainty*. Model uncertainty implies that the network output

[3]Gross model misspecification error (arising *e.g.* when not enough hidden units are used or when gross overfitting has occurred) is not considered, *i.e.* the network architecture is assumed to be optimised.

should be viewed as probabilistic. The output distribution can be considered again Gaussian[4] or Student's t [16, 19, 30] with mean equal to the most probable output and variance $\sigma_m^2(\mathbf{x})$, referred to as *model uncertainty variance.*

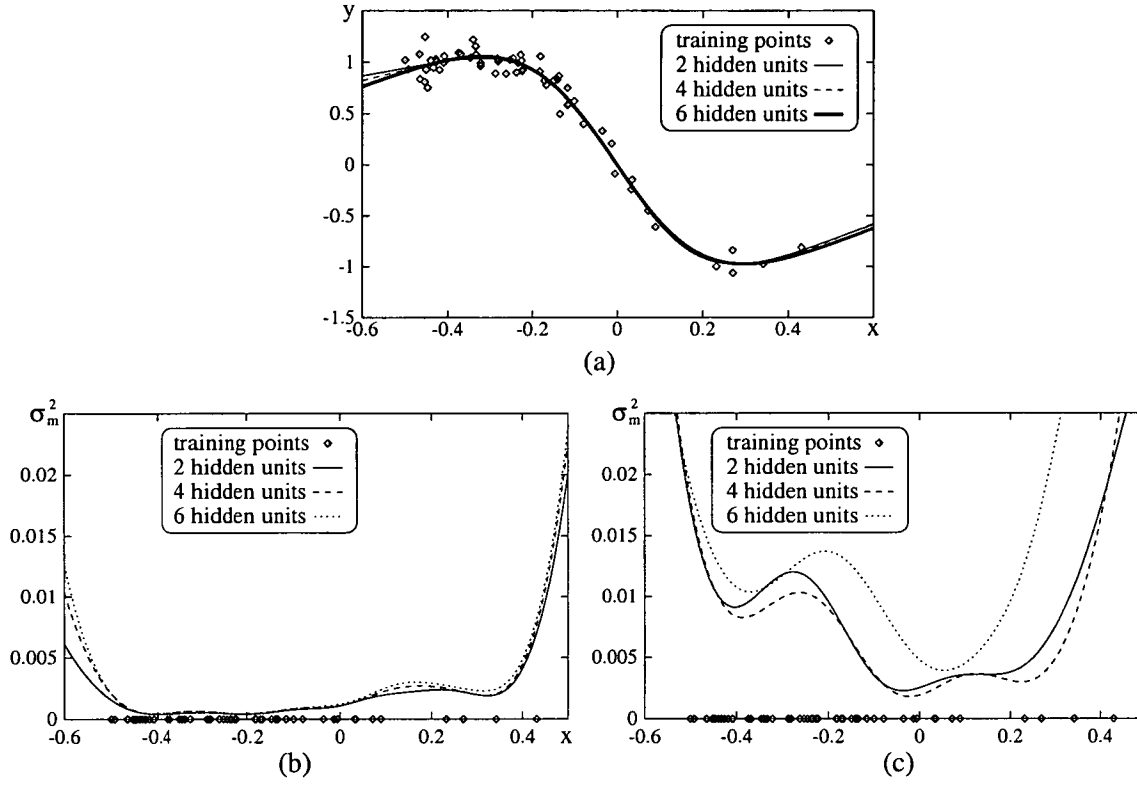


Figure 2.2: *The output of an MLP trained on an one-dimensional artificial problem (shown in (a)) and the associated model uncertainty variance estimates using: (b) the approximate delta method and (c) the bootstrap method. Networks of 2, 4 and 6 hidden units have been used to model the regression. The methods used to obtain the model uncertainty estimates are described in the following sections.*

Effectively, model uncertainty is due to the difference $y(\mathbf{x}) - f(\mathbf{x})$, *i.e.* the uncertainty in learning the true regression [19]. For MLPs, if the number of hidden units is assumed to be optimum, the error $y(\mathbf{x}) - f(\mathbf{x})$ can be attributed to error in the inferred weight values $\hat{\mathbf{w}} - \dot{\mathbf{w}}$ (*e.g.* [23]). Therefore, in this case, model uncertainty is mainly due to weight value misspecification. As explained in [18] weight value estimates are less accurate in low data density regions. Therefore, model uncertainty should increase in regions of small data density.

In practice, the estimate of model uncertainty variance increases in low PDF regions as the

[4] A theoretical justification for the use of the Gaussian distribution is given by the *central limit theorem* [34] according to which the mean of M random variables tends to be normally distributed, in the limit as M tends to infinity.

number of hidden units increases. This is because, for the same amount of data, the number of well determined weights (*i.e.* weights whose value is determined by the data) decreases as the total number of weights increases. This is illustrated in fig. 2.2 using one-dimensional artificial data and networks of 2, 4 and 6 hidden units. Although the output of the three networks is very similar (see fig. 2.2 (a)), the model uncertainty variance estimates, shown in fig. 2.2 (b) and (c), are generally higher for larger number of hidden units. This effect is particularly evident in regions of low data density. This issue is not investigated here in depth as it is assumed that the most appropriate network architecture has already been determined by examining generalization performance. The methods used to estimate model uncertainty variance will be described in detail in section 2.4.

2.3.3 Total prediction uncertainty

In this work the Gaussian distribution is used to model both data noise and model uncertainty distributions for reasons of simplicity and uniformity. As mentioned previously, a Student's t distribution should be used in a real system implementation to obtain more accurate results for finite data sets. The one-dimensional Gaussian distribution is given by

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \tag{2.8}$$

where μ is the mean and σ^2 is the variance of the distribution. For convenience, a Gaussian distribution with mean μ and variance σ^2 is denoted by $N(\mu, \sigma^2)$.

Since both sources of uncertainty are modelled as Gaussian distributions, uncertainty can be estimated by estimating the variances σ_ν^2 and σ_m^2 of the two distributions. Because $\hat{\mathbf{w}} - \dot{\mathbf{w}}$ (the difference between estimated and optimum weight values) and e (the data noise component) are statistically independent [19, 23, 67] the total prediction variance is given by the sum of their variances

$$\sigma_{\text{TOTAL}}^2 = \sigma_\nu^2 + \sigma_m^2 \tag{2.9}$$

In section 2.4 various methods for estimating σ_ν^2 and σ_m^2 are presented.

In certain applications, such as electric load forecasting [14] or engine calibration [68] for which a relatively large data set is available, estimation of model uncertainty variance is some-

times omitted. This is because it can be assumed that the high data density ensures that uncertainty due to model misspecification is negligible compared to uncertainty due to data noise. Moreover, the nature of these applications is such that all new input patterns must originate from regions represented in the training data. However, in problems such as the curl task, where the data set is limited and the system operator can manually vary the input parameters, it is essential to estimate both σ_ν^2 and σ_m^2 in order to obtain a complete confidence estimate.

2.3.4 Uncertainty in curl prediction

As far as the curl data are concerned, it is known that the targets are corrupted by errors [21, 22]. In [21] it was found that there are few or no gross outliers in the curl data. However, that work used an older data set which is not used in this thesis.

Moreover, the curl data are very sparse. In this thesis, two curl sets are used for training, comprising 369 and 448 patterns and having 8 and 11 inputs respectively. These are particularly sparse data sets, because the quantity of training data needed to describe a mapping grows exponentially with the number of input parameters. This is known as the *curse of dimensionality* [69]. Therefore, estimation of both data noise and model uncertainty is crucial for the curl estimation system.

In this work, data noise and model uncertainty are assumed to be governed by a Gaussian distribution. In practice, a Student's t distribution should be used. However, even a t distribution may not describe accurately the complex processes involved in paper manufacture. There exist advanced methods for computing the actual output distribution without the need to assume a distribution form. These include the *hybrid Monte Carlo* techniques described in section 2.4.3 or advanced bootstrap algorithms mentioned in section 2.4.4. However, it will be shown that such methods present a number of problems from a practical point of view. Therefore, the use of a Gaussian or Student's t distribution is a satisfactory alternative for the output distribution of the curl estimator.

Likewise, data noise variance is modelled by a Gaussian (or Student's t) distribution of constant or input-dependent variance. This is an improvement over the initial modelling attempts [21, 22] in which only constant data noise variance was considered.

Finally, it is known that the value of some of the input parameters may suffer uncertainty due to inaccurate measurement. For example, *porosity* is measured by a human operator. All other

parameters used for curl prediction are either set by an operator or measured automatically by a computer (see appendix B). An outlier in the explanatory (input) parameters or *high-leverage* point can significantly deteriorate generalization performance in linear regression models [21, 70]. This issue was addressed in [21] where it was found that high-leverage points are not a serious problem in MLP regression as they only cause local overfitting problems. Therefore, the assumption that input values are correct is not likely to significantly influence confidence estimation for the curl task.

2.4 Confidence estimation methods for ANNs

In this section the considered confidence estimation techniques are presented. The focus is on the algorithmic and qualitative differences and merits of each technique. Other confidence estimation techniques, which were not eventually considered as candidates, are also briefly presented and the reasons for their exclusion are explained.

A fairly common approach to confidence estimation is to treat the noise inherent to the data as Gaussian with a constant variance σ_ν^2 and use *Maximum Likelihood (ML)* and the *delta* method (a variant of *linearization* [15]) to obtain estimates of the data noise variance σ_ν^2 and model uncertainty variance σ_m^2 respectively [20]. In [27] a method for obtaining an input-dependent estimate of σ_ν^2 using ML is presented. The traditional network architecture is extended and a new set of hidden units is used to compute $\hat{\sigma}_\nu^2(\mathbf{x})$, the network estimate for data noise variance. This may lead to considerable improvement in real applications, where the data noise level is not constant. The so-called *Evidence Framework (EF)* approximation to the Bayesian approach can be used to obtain predictions and uncertainty measures treating data noise variance as constant [29, 30, 32] or allowing it to be a function of the inputs [17]. EF uses a Gaussian approximation to the posterior to avoid time-consuming Monte-Carlo integration over the weight space, required by the exact Bayesian approach [71]. Finally, the *bootstrap* technique can be used to obtain estimates of both regression and model uncertainty variance [33]. In [19] the bootstrap algorithm is extended and an input-dependent estimate of data noise variance is obtained as well.

In the rest of this section, the methods are described in more detail.

2.4.1 Non-Linear regression theory

Uncertainty associated with the parameters of a non-linear model can be estimated using a method known as *linearization* [15, 67]. Because the number of model parameters in neural networks is very large, estimating uncertainty for the network parameters is analytically intractable. However, the same technique can be adapted to obtain estimates of the uncertainty associated with the network output [16, 25].

Linearization is based on the assumption that the non-linear function can be approximated by an affine or linear function in a small neighbourhood of the weight space around the estimated weight vector. Under this assumption, the standard linear regression theory result [72] can be used

$$\hat{\sigma}_m^2(\mathbf{x}) = \mathbf{g}^T(\mathbf{x})\mathbf{V}^{-1}\mathbf{g}(\mathbf{x}) \tag{2.10}$$

where $\mathbf{g}(\mathbf{x})$ is the gradient vector of the output given by

$$\mathbf{g}(\mathbf{x}) = \left[\frac{\partial y(\mathbf{x})}{\partial \hat{w}_1}, \frac{\partial y(\mathbf{x})}{\partial \hat{w}_2}, ..., \frac{\partial y(\mathbf{x})}{\partial \hat{w}_W}\right] \tag{2.11}$$

and $\mathbf{V}$ is the covariance matrix of the weights.

Three alternative approximations to the covariance matrix can be used [15]

$$\mathbf{V}_1 = \hat{\sigma}_\nu^2(\mathbf{J}^T\mathbf{J})^{-1} \tag{2.12}$$

$$\mathbf{V}_2 = \hat{\sigma}_\nu^2\mathbf{H}^{-1} \tag{2.13}$$

$$\mathbf{V}_3 = \hat{\sigma}_\nu^2\mathbf{H}^{-1}(\mathbf{J}^T\mathbf{J})\mathbf{H}^{-1} \tag{2.14}$$

where $\hat{\sigma}_\nu^2$ is a constant estimate of data noise variance, $\mathbf{J}$ the *Jacobian* matrix of first order derivatives of the output with respect to the weights, and $\mathbf{H}$ the exact *Hessian* matrix of second order derivatives of the error function with respect to the weights.

Equation (2.10) using approximation (2.13) is sometimes referred to as the *delta* method for model uncertainty variance estimation [33, 73]. Equation (2.13) corresponds to the constant noise variance case. If the noise variance is treated as a function of the inputs and weight decay regularization is used, the covariance matrix can be approximated by the exact Hessian which

is given in this case by [17]

$$\mathbf{H} = \sum_{n=1}^{N} \frac{1}{\hat{\sigma}_\nu^2(\mathbf{x}^n)} \frac{\partial^2 E^n}{\partial \mathbf{w}^2} + \alpha \mathbf{I} \tag{2.15}$$

where $\mathbf{I}$ is the unitary matrix of size $W \times W$ and E^n is the least squares error for data point $\mathbf{x}^n$ given by

$$E^n = \frac{1}{2}[t^n - y(\mathbf{x}; \hat{\mathbf{w}})^n]^2 \tag{2.16}$$

In the following sections it will be shown how a MLP can be used to obtain $\hat{\sigma}_\nu^2(\mathbf{x}^n)$, an input-dependent estimate of data noise variance. In the rest of the thesis the delta estimator using the exact Hessian is referred to as the *exact delta* method.

Approximation (2.12) corresponds to the outer-product approximation to the Hessian [73]. Using some simple matrix algebra [74] it can be shown that the outer-product approximation to the Hessian using weight decay regularization and input-dependent data noise variance is given by

$$\tilde{\mathbf{H}} = \sum_{n=1}^{N} \frac{1}{\hat{\sigma}_\nu^2(\mathbf{x}^n)} \mathbf{g}(\mathbf{x}^n)\mathbf{g}^T(\mathbf{x}^n) + \alpha \mathbf{I} \tag{2.17}$$

In the rest of this thesis $\tilde{\mathbf{H}}$ is referred to as the approximate Hessian. Equation (2.10) using the approximate Hessian is referred to as the *approximate delta* method.

Equation (2.10) with the approximation of eq. (2.14) is equivalent to the so-called *sandwich* estimator [33,73]. This variation is not considered here since it was shown to give generally poorer or similar results to the other two while being computationally more complex [15,73]. All terms are evaluated at the estimated weight values $\hat{\mathbf{w}}$. Note that in the delta method the σ_m^2 estimate depends on the σ_ν^2 estimate.

Calculation of the exact Hessian matrix is straightforward and can be performed using an algorithm similar to back-propagation [75,76]. However, some accuracy is lost in the inversion of the Hessian matrix which is of size $W \times W$. The inverse of a matrix can be computed using numerical methods such as *Gauss-Jordan elimination* [50,77], *LU* decomposition [77–79], or *singular value decomposition* (SVD) [78,80–82]. In this work SVD has been used for matrix inversion. This technique is considered the most computationally robust and efficient amongst

available numerical methods for matrix inversion [50].

It must be noted that methods based on evaluation of the Hessian result in approximate confidence intervals [15]. Approximate intervals may not contain the true value with the theoretical probability. Finally, approximating the covariance matrix with the exact or approximate Hessian assumes that the model is correct [15, 73]. Thus, performance of the delta method can be expected to degrade when the model is not correct.

2.4.2 Maximum likelihood

Maximum likelihood (ML) is a common approach to estimating statistical parameters from a given data set [83,84]. In a neural network context, ML can be used to train the network (*i.e.* estimate weight values from the data) [34]. Additionally, ML methods can be used to estimate data noise variance either as a constant [23, 84, 85] or a function of the inputs [26–28, 86].

ML methods require that an analytical, parametric form is assumed for the *conditional target distribution* $p(t|\mathbf{x})$, *i.e.* the probability distribution of the targets given the inputs. Subsequently, ML seeks to maximise the likelihood of the data [34]. In practice, it is more convenient to consider the equivalent procedure of minimising the negative log-likelihood or *total error* function. Thus, ML makes predictions based on a single-best set of weight values. It can be shown that, if the errors are assumed to be normally distributed, the least squares estimator coincides with the ML estimator [84, 87]. Therefore, for a neural network employed to solve a regression task with normally distributed data errors the ML error function with weight decay regularization is given by eq. (2.7), which is repeated here for convenience

$$C(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}[t^n - y(\mathbf{x}^n;\mathbf{w})]^2 + \frac{\alpha_w}{2}\sum_{i=1}^{W} w_i^2 \tag{2.18}$$

Implicit in this equation is the assumption that the noise in the data is constant over input space. Moreover, data noise variance is considered constant during training and is therefore omitted from this equation as it does not influence the minimisation procedure. The ML estimate of data noise variance as a constant [84, 85] is given by

$$\hat{\sigma}_\nu^2 = \frac{1}{N}\sum_{n=1}^{N}[t^n - y(\mathbf{x}^n;\hat{\mathbf{w}})]^2 \tag{2.19}$$

where $\hat{\mathbf{w}}$ is the vector of weight values estimated by minimising eq. 2.18.

An input-dependent data noise variance estimate can be obtained by training a second neural network using the squared residuals $[t^n - y(\mathbf{x}^n; \mathbf{w})]^2$ as targets. This can be achieved by training a supplementary network *after* training the regression network is completed [86]. However, it is advantageous to treat the problem in an inter-linked manner and learn both regression and noise variance at the same time [26, 27]. This can be achieved using the *augmented* network architecture of fig. 2.3.

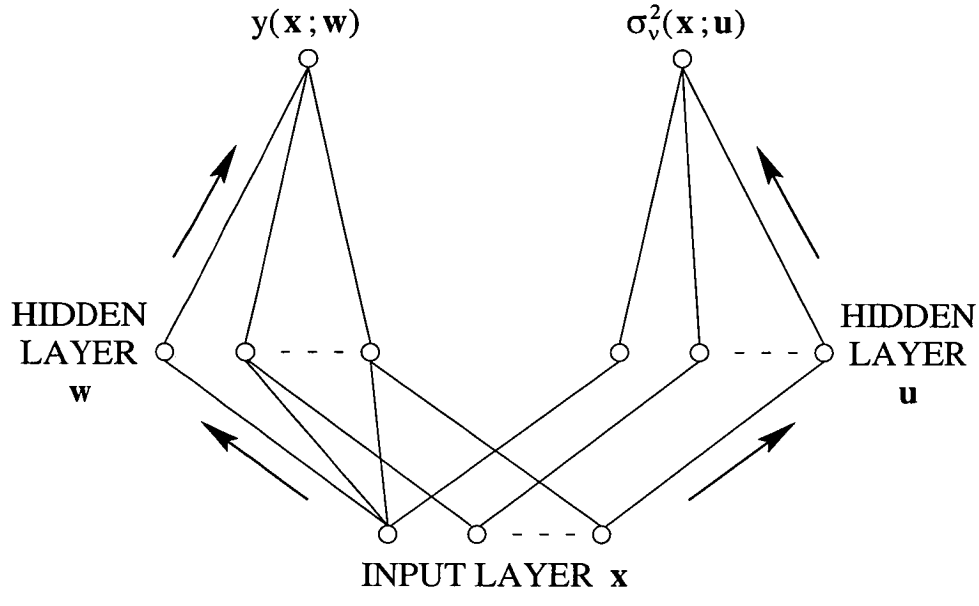


Figure 2.3: *The augmented network architecture for obtaining an input dependent data noise variance estimate. In the general case the input and hidden layers are fully connected. Not all possible connections are shown in the figure for clarity.*

The total error with weight decay regularization is now given by [27]

$$C(\mathbf{w};\mathbf{u}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\frac{[t^n - y(\mathbf{x}^n;\mathbf{w})]^2}{\sigma_\nu^2(\mathbf{x}^n;\mathbf{u})} + \ln\sigma_\nu^2(\mathbf{x}^n;\mathbf{u})\right\} + \frac{\alpha_w}{2}\sum_{i=1}^{W} w_i^2 + \frac{\alpha_u}{2}\sum_{i=1}^{U} u_i^2 \quad (2.20)$$

where $\mathbf{w}$ and $\mathbf{u}$ are the weight vectors for the regression and variance subnetworks respectively, α_w and α_u are the regularization parameters for the regression and variance weights respectively and U is the total number of weight connections in the data noise variance subnetwork. The data noise variance estimate now enters the calculations as it is no longer constant. It is convenient to use an exponential activation function for the noise variance output unit so that the variance can only take positive values. Error $C(\mathbf{w};\mathbf{u})$ is minimised jointly for $y(\mathbf{x};\mathbf{w})$ and $\sigma_\nu^2(\mathbf{x};\mathbf{u})$ to obtain the estimated weight values $\hat{\mathbf{w}}$ and $\hat{\mathbf{u}}$. This approach can be extended for

the case of multi-modal distributions (*e.g.* distributions arising in inverse problems) [88] or for multi-dimensional targets [28].

The effect of learning the regression and noise variance simultaneously is that the learning rate for the regression weights is inversely proportional to the noise variance estimate [27]. Thus, the weight values are determined primarily by low noise regions and a more accurate regression estimate is obtained in such regions. Conversely, regions of high noise have less influence upon the network weights. This is not the case if, first, regression is estimated by minimising eq. (2.18) and then an input-dependent variance estimate is obtained by training a supplementary neural network.

ML is a simple, fast and easy to use technique. The main disadvantage of this approach is that, as the regression estimate $y(\mathbf{x}; \hat{\mathbf{w}})$ fits some of the data noise, the obtained data noise variance estimate is biased. Thus, data noise may be underestimated, particularly in regions of low data density where overfitting is more probable [17]. A possible remedy is to reserve a fraction of the training data and use it for training the variance network exclusively. This will partly remove the bias, as the regression network has not seen these data. However, training must now be performed in two (or three) separate phases increasing algorithm complexity (see *e.g.* [27]). Moreover, partitioning the data set may lead to poor initial regression estimate.

In this work the two interconnected networks are trained in tandem using the same training data for regression and variance learning. However, variance learning is initially delayed for a number of training epochs until a sufficient level of regression learning has been achieved. This prevents the learning process from being stalled in local minima that correspond to highly erroneous solutions [27].

A number of similar approaches to uncertainty estimation adopt no assumption about the error distribution and compute an input-dependent error estimate by training a supplementary neural network to predict the squared residuals [14, 68]. In theory the error estimate thus obtained is not the variance of a Gaussian distribution. However, as mentioned in [34], these methods are effectively equivalent and suffer the same problems as ML methods.

If an ML approach is used to estimate regression and data noise variance, model uncertainty variance can be estimated using the exact or approximate delta methods, described in section 2.4.1.

2.4.3 Bayesian methods

Unlike ML, in the Bayesian approach predictions are based on the entire output probability distribution. Over the last years Bayesian methods have been applied to ANN training and inference and a variety of algorithms have been developed [29, 30, 32, 41, 71, 89–91].

In Bayesian analysis, the output distribution is governed by the *prior* weight distribution, which corresponds to any knowledge about the nature of the problem prior to the arrival of the data, and the *posterior* weight distribution, *i.e.* the weight distribution after the data have been taken into account. In a neural network context, prior knowledge is expressed by the regularization term. For example, weight decay corresponds to the prior knowledge that the regression function is smooth and weights must therefore take small values. The Bayesian approach can also be used to estimate data noise variance. In this context, the main advantage of the Bayesian approach over ML is that it results in an unbiased data noise variance estimate.

The exact Bayesian analysis [71, 92, 93] involves calculation of integrals over weight space which can not be performed analytically but only using iterative random sampling procedures. Such methods are called *Monte Carlo* techniques (*e.g.* the Metropolis algorithm [94] or Gibbs sampling [95]) and can be used to perform random sampling from multi-dimensional probability distributions. In the case of Bayesian analysis, a modified algorithm known as *hybrid* Monte Carlo [96, 97] has been proposed for integration over weight space [71, 92]. Hybrid Monte Carlo incorporates information concerning the gradient of $p(\mathbf{w}, D)$ in the Monte Carlo procedure. The technique results in a random sample set of weight values from the distribution $p(\mathbf{w}|D)$. This set can be used to obtain network predictions that represent samples from the output distribution $p(y|\mathbf{x}, D)$. Subsequently, these samples can be used to obtain a most probable prediction and an associated confidence interval.

Although, at present, exact Bayesian algorithms using hybrid Monte Carlo treat data noise variance as constant, in principle, it is possible to extend this approach to the input-dependent noise variance case [71]. The disadvantage of this approach is the lack of any suitable terminating criterion for the Monte Carlo sampling and the significant amount of computations involved. Therefore, the exact Bayesian approach is too computationally demanding and exceeds the practical limitations of the curl task.

These problems are remedied to a certain degree if the posterior weight distribution is approximated by a Gaussian [29, 32, 41, 90]. It is known that this approximation will not be good for

large networks and/or small data sets for which the ratio N/W has a small value [30]. However, the approximate Bayesian approach is very useful in practice as it allows the Bayesian integrals to be computed analytically saving a considerable amount of computation.

An interesting practical issue for such methods is the handling of *hyperparameters*. Hyperparameters are parameters that control the distribution of model parameters such as weights and biases. For example, the regularization parameters can be viewed as hyperparameters because they control the prior distribution of the weights. There are two alternative methods for performing the analysis. In the so-called Evidence Framework (EF) [30, 41] the value of the *hyperparameters* is continuously estimated from the data and updated during network training. Alternatively, the hyperparameters can be eliminated by analytical integration [29, 32, 98]. Although, exact integration is theoretically superior and the EF procedure introduces approximation error [99], it has been shown that, in practice, EF yields superior results [98].

Another open issue is the choice of priors. The most common form is a *quadratic* prior (*e.g.* [41, 90]) that corresponds to the common weight decay term in the error function. Other alternatives include entropic [100] or Laplace priors [32, 101]. A more general, adaptive prior in the form of a mixture of Gaussians is proposed in [102]. A comparison of some priors can be found in [29].

In the EF algorithm with constant noise level, both the regularization parameter α and data noise variance estimate $\beta = 1/\sigma_\nu^2$ are treated as hyperparameters. The total error function to be minimised is given by

$$S(\mathbf{w}) = \beta E_D + \alpha E_W \tag{2.21}$$

where E_D, E_W are the sum-of-squares and weight decay errors given as

$$E_D = \frac{1}{2}\sum_{n=1}^{N}[t^n - y(\mathbf{x}^n; \mathbf{w})]^2 \tag{2.22}$$

$$E_W = \frac{1}{2}\sum_{i=1}^{W} w_i^2 \tag{2.23}$$

The difference of this procedure compared to ML is that the values of β and α are continuously

updated during training using the following equations [30,41]:

$$\alpha^{\text{new}} = \gamma/2E_W \tag{2.24}$$

$$\beta^{\text{new}} = (N-\gamma)/2E_D \tag{2.25}$$

where γ is a measure of the effective number of weights which are well determined by the data. The number of well determined weights is given by

$$\gamma = \sum_{i=1}^{W} \frac{\lambda_i}{\lambda_i + \alpha_w} \tag{2.26}$$

where λ_i, $i = 1, ..., W$ are the eigenvalues of the Hessian matrix of error E_D with respect to weights $\mathbf{w}$. Thus, the constant data noise variance estimate of the Bayesian method can be computed as

$$\sigma_\nu^2 = \frac{1}{N-\gamma} \sum_{n=1}^{N} [t^n - y(\mathbf{x}^n; \hat{\mathbf{w}})]^2 \tag{2.27}$$

This result should be contrasted with eq. (2.19) which gives the biased ML estimate. Unlike ML, in the Bayesian approach the averaging takes into account the number of well-determined weights (*i.e.* "used" degrees of freedom) and the divider is set to $N - \gamma$ instead of N.

The approximate Bayesian method has been extended for the case of input-dependent noise variance [17,91]. Like ML, the algorithm makes use of the augmented network architecture (see fig. 2.3). Optimisation is now performed separately for the regression and variance networks. First, weights $\mathbf{w}$ are optimised for the current value of $\mathbf{u}$ by minimising the following error function:

$$C_r(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \frac{[t^n - y(\mathbf{x}^n; \mathbf{w})]^2}{\sigma_\nu^2(\mathbf{x}^n; \mathbf{u})} + \frac{\alpha_w}{2} \sum_{i=1}^{W} w_i^2 \tag{2.28}$$

Subsequently, the noise variance weights $\mathbf{u}$ are optimised for the current value of $\mathbf{w}$ by minimising

$$C_\nu(\mathbf{u}) = \frac{1}{2} \sum_{n=1}^{N} \frac{[t^n - y(\mathbf{x}^n; \mathbf{w})]^2}{\sigma_\nu^2(\mathbf{x}^n; \mathbf{u})} + \frac{1}{2} \sum_{n=1}^{N} \ln \sigma_\nu^2(\mathbf{x}^n; \mathbf{u}) + \frac{1}{2} \ln |\mathbf{H}| + \frac{\alpha_u}{2} \sum_{i=1}^{U} u_i^2 \tag{2.29}$$

where $\mathbf{H}$ is the Hessian matrix of error $C_r(\mathbf{w})$ with respect to weights $\mathbf{w}$ and $|\mathbf{H}|$ is the determ-

inant of $\mathbf{H}$.

In practical terms, this approach differs from the corresponding ML approach in two ways. First, the two sets of weights ($\mathbf{w}$ and $\mathbf{u}$) are optimised separately rather than jointly. Second, the term $\frac{1}{2} \ln |\mathbf{H}|$ is introduced in the error function used to estimate data noise variance (eq. (2.29)). Note that otherwise this equation is identical to eq. (2.20) used in ML learning. The term $\frac{1}{2} \ln |\mathbf{H}|$ is due to *marginalization* over weights $\mathbf{w}$. This process removes the dependence of $\hat{\mathbf{u}}$ on the regression estimate $y(\mathbf{x}^n; \hat{\mathbf{w}})$ and therefore leads to an unbiased data noise variance estimate. The main disadvantage of the approximate Bayesian method compared to ML is the increased amount of computations required due to the frequent calculation of the Hessian matrix.

If the approximate Bayesian method is used, model uncertainty variance can be estimated using the exact or approximate delta methods as in the case of ML. However, since the delta estimate of model uncertainty variance depends on the noise variance estimate, a more accurate σ_m^2 estimate will be obtained as the data noise variance estimate is now unbiased.

2.4.4 Bootstrap method

The bootstrap technique is a data based simulation method for estimating the standard error of a statistical parameter [33]. In a neural network context, bootstrap methods can be used to obtain estimates of the regression and model uncertainty variance. The technique is based on random sampling from the available data set to create a number of resampled, bootstrap sets. The available data set can be viewed as a random sample from the true data distribution. Bootstrap sampling gauges the true distribution by resampling from the available data. In other words, resampling the available data is considered equivalent to sampling from the original data distribution which is naturally unknown.

There are two alternative bootstrap sampling algorithms. *Pairs sampling* is based on *resampling with replacement* the available data set and training a committee of networks, each on a resampled instance of the original data set. Resampling with replacement means that after a data pair is chosen, it is replaced in the original set. As the size of the resampled sets equals the size of the original set N, a resampled set may contain some pairs more than once while other pairs may not be present at all.

An alternative method called *residual sampling* requires a trained network model to exist be-

fore bootstrap sampling is performed. Sampling is performed by randomly choosing (using resampling with replacement) from the set of model residuals. Bootstrap replicate sets are created by adding the sampled residual errors to the model's predictions. The set of predictors (input variables) remains the same for all bootstrap replicates. This scheme conditions the results on the particular set of predictors under the assumption that the initial model is correct [1, 33]. This is unlikely to be desirable in neural network applications [73] in which the predictions and confidence estimates must be valid for new, unseen inputs. Since residual sampling also has the additional requirement of a pre-existing "correct" model, it was concluded that pairs sampling is more appropriate for ANN applications such as curl estimation.

The details of the pairs sampling algorithm are given below.

1. Generate B bootstrap replicate sets $\mathcal{D}_i^*$, $i = 1, ..., B$ of the original set $\mathcal{D}$ using resampling with replacement. A resampled data set has the form $\{(\mathbf{x}^{n*}, t^{n*})\}$, where $n*$ can take any value in $[1, N]$ with probability $1/N$. For every replicate set, the remaining fraction or out-of-sample set is denoted $\mathcal{D} - \mathcal{D}_i^*$.

2. For each i train a network of the same architecture on $\mathcal{D}_i^*$. If required, the remaining fraction $\mathcal{D} - \mathcal{D}_i^*$ can be used for validation.

3. Obtain the bootstrap committee's regression and σ_m^2 estimates by

$$y(\mathbf{x}) = \sum_{i=1}^{B} y_i(\mathbf{x})/B \tag{2.30}$$

$$\hat{\sigma}_m^2(\mathbf{x}) = \sum_{i=1}^{B} [y_i(\mathbf{x}) - y(\mathbf{x})]^2/(B-1) \tag{2.31}$$

   where $y_i(\mathbf{x})$ is the prediction of network i.

Note that, effectively, the bootstrap estimate of model uncertainty variance refers to an architecture rather than to a particular single network.

After training is completed, a constant data noise variance estimate can be obtained using eq. (2.19). This estimate will be biased. However, an unbiased data noise variance estimate can be obtained by taking advantage of the bootstrap training algorithm. Instead of using the model's regression estimate $y(\mathbf{x}; \hat{\mathbf{w}})$ as in eq. (2.19), the residuals can be calculated using the predictions of each network on the corresponding out-of-sample set. Therefore, the predictions

of the individual networks on the out-of-sample sets $\mathcal{D} - \mathcal{D}_i^*$ are averaged to obtain an unbiased regression estimate $\hat{y}_{\text{unbiased}}$ as follows

$$\hat{y}_{\text{unbiased}}(\mathbf{x}) = \frac{\sum_{i=1}^{B} v_i(\mathbf{x}) \cdot y_i(\mathbf{x})}{\sum_{i=1}^{B} v_i(\mathbf{x})} \tag{2.32}$$

where $v_i(\mathbf{x})$ is one if pattern $\mathbf{x}$ is in out-of-sample set $\mathcal{D} - \mathcal{D}_i^*$ and zero otherwise. The regression estimate $\hat{y}_{\text{unbiased}}(\mathbf{x})$ is unbiased because it is computed using data patterns that were not used to train the networks. Subsequently, an unbiased constant data noise variance estimate can be obtained by

$$\hat{\sigma}_\nu^2 = \frac{1}{N} \sum_{n=1}^{N} \max \left\{ [t^n - \hat{y}_{\text{unbiased}}(\mathbf{x}^n)]^2 - \hat{\sigma}_m^2(\mathbf{x}^n), 0 \right\} \tag{2.33}$$

Note that the estimate of model uncertainty variance $\hat{\sigma}_m^2$ is subtracted from the residual so that it does not influence noise variance estimation. This scheme will only have significant effect in low input data density regions where σ_m^2 may be comparable to σ_ν^2.

The same approach can be used to compute an unbiased, input-dependent noise variance estimate [19]. First, the unbiased regression estimate is used to compute an unbiased set of the model's residuals on the training set. As previously, the adapted residual for pattern $(\mathbf{x}^n, t^n)$ is given as

$$r^2(\mathbf{x}^n) = \max \left\{ [t^n - \hat{y}_{\text{unbiased}}(\mathbf{x}^n)]^2 - \hat{\sigma}_m^2(\mathbf{x}^n), 0 \right\} \tag{2.34}$$

Subsequently, a supplementary network is trained on set $\left\{ (\mathbf{x}^n, r^2(\mathbf{x}^n)) \right\}$, $n = 1, ..., N$ using the following error function:

$$S(\mathbf{u}) = \frac{1}{2} \sum_{n=1}^{N} \left\{ \frac{r^2(\mathbf{x}^n)}{\sigma_\nu^2(\mathbf{x}^n; \mathbf{u})} + \ln \sigma_\nu^2(\mathbf{x}^n; \mathbf{u}) \right\} + \frac{\alpha_u}{2} \sum_{i=1}^{U} u_i^2 \tag{2.35}$$

Note that this equation is equivalent to eq. (2.20) with the regression estimate treated as constant. A similar approach to obtaining an unbiased estimate of noise variance is used in [103] for a classification task, employing a scheme similar to cross validation - instead of bootstrap sampling - to partition the training set.

In practice, the assumption that the output distribution is Gaussian will be inaccurate for small data sets. Therefore, the bootstrap technique leads to the formulation of approximate confidence

intervals as well. In general, there exist advanced bootstrap methods that can be used to obtain confidence intervals that do not have a Gaussian distribution (*e.g.* the bootstrap *percentile* intervals or the BC_α method [33]). However, the number of bootstrap replications ($B \geq 500$ or 1000) required by such methods is prohibiting for the category of ANN applications studied here, due to the long query times involved.

As explained in sections 2.4.2 and 2.4.3, both the ML and approximate Bayesian methods employ the augmented network architecture (see fig. 2.3) for making predictions under the input-dependent data noise variance assumption. In this thesis, methods that use the augmented architecture are called *augmented network* methods. In contrast, the bootstrap regression estimate is obtained by training a committee of networks of a conventional MLP architecture. Both the augmented and bootstrap methods can be used to obtain an input-dependent data noise variance estimate. However, as mentioned in section 2.4.2, in augmented network methods the learning rate of the regression weights is inversely proportional to the current data noise variance estimate. This leads to a qualitatively different regression estimate compared to the bootstrap method in which regression estimation is performed under the (implicit) assumption of constant data noise variance.

2.4.5 Other approaches to confidence estimation

Over the last few years a *Gaussian process* [104] approach to neural networks has emerged [105–108]. This approach employs a Gaussian *prior over functions* to describe a modelling problem. Neural networks can be viewed as a special case of Gaussian processes. It has been shown that the properties of a MLP with one hidden layer converge to those of a Gaussian process as the number of hidden units tends to infinity [71, 109].

Gaussian processes is an open research area. Although this approach may seem promising it presents a number of practical disadvantages. First, regression using Gaussian processes requires the inversion of a matrix of size $N \times N$. This means that for large data sets approximate inversion methods must be used and the amount of computations required is significantly large. Moreover, matrix inversion may be non-stable and prone to numerical inaccuracies. Finally, comparison studies of Gaussian processes and neural networks have shown that the former may not always lead to a significant improvement [68, 105]. For these reasons, the Gaussian process approach is not considered as a candidate for the curl task.

2.5 Committees of networks and uncertainty

Combining the predictions of a number of individual networks can lead to significant improvement in generalization performance [110, 111]. This approach is generally known as *committees or ensembles of networks*. The principle behind network committees is that each individual network learns a slightly different representation of the data. Therefore, because the errors have a different distribution from network to network, a proportion of the error is eliminated by combining the predictions [112].

A variety of methods for training and combining the predictions of individual networks have been proposed [113–117]. In this context, the bootstrap technique can be viewed as a committee formation technique. The most simple approach is to train each network on all the available data and average the predictions of individual networks $y_i(\mathbf{x})$ (*e.g.* [118])

$$y_{\mathrm{COM}}(\mathbf{x}) = \sum_{i=1}^{M} y_i(\mathbf{x})/M \tag{2.36}$$

where the sum is over each individual network and M is the total number of networks in the committee.

$$\begin{aligned}
&<y_i^2>-<y_i>^2 = <y_i^2>-<y_i><y_i> \Leftrightarrow \\
\Leftrightarrow\ &<y_i^2>-<y_i>^2 = <y_i^2>-<y_i><y_i>+<y_i><y_i>-<y_i><y_i> \Leftrightarrow \\
\Leftrightarrow\ &<y_i^2>-<y_i>^2 = <y_i^2>-2<y_i>y_{\mathrm{COM}} + y_{\mathrm{COM}}^2 \Leftrightarrow \\
\Leftrightarrow\ &<y_i^2>-<y_i>^2 = \sum_{i=1}^{M}\frac{y_i^2}{M} - 2y_{\mathrm{COM}}\sum_{i=1}^{M}\frac{y_i}{M} + y_{\mathrm{COM}}^2\sum_{i=1}^{M}\frac{1}{M} \Leftrightarrow \\
\Leftrightarrow\ &<y_i^2>-<y_i>^2 = \sum_{i=1}^{M}\frac{y_i^2 - 2y_i y_{\mathrm{COM}} + y_{\mathrm{COM}}^2}{M} \Leftrightarrow \\
\Leftrightarrow\ &<y_i^2>-<y_i>^2 = \sum_{i=1}^{M}\frac{[y_i - y_{\mathrm{COM}}]^2}{M}
\end{aligned}$$

Table 2.1: *Details of the mathematical derivation of eq. (2.38).*

When networks trained using ML or the approximate Bayesian approach are used to form

a committee, the total variance associated with the committee prediction is not equal to the average of the variances of the individual networks [29, 90]. The total variance is given by

$$\hat{\sigma}^2_{\text{TOTAL}} = <\hat{\sigma}_i^2> + \{<y_i^2> - <y_i>^2\} \tag{2.37}$$

where $\hat{\sigma}_i^2$ is the total prediction variance for network i.

Using eq. (2.36) and some algebraic manipulation (see table 2.1) a more familiar expression can be obtained for the additional term $\{<y_i^2> - <y_i>^2\}$ in eq. (2.37). This term expresses the uncertainty due to the difference between the predictions of individual committee members [90]. Using the result in table 2.1, the estimate of committee uncertainty $\hat{\sigma}^2_{\text{COM}}(\mathbf{x})$ can be written as

$$\hat{\sigma}^2_{\text{COM}}(\mathbf{x}) \equiv \{<y_i^2> - <y_i>^2\} = \sum_{i=1}^{M} \frac{[y_i(\mathbf{x}) - y_{\text{COM}}(\mathbf{x})]^2}{M} \tag{2.38}$$

Note the similarity of this expression and eq. (2.31) that gives the estimate of model uncertainty variance in the bootstrap technique. In fact these two equations only differ in the denominator. In the case of the bootstrap estimate, the denominator is set to $M-1$ instead of M. As explained in sections 2.4.2 and 2.4.3, dividing by the number of examples in the data set (in this case the number of individual network predictions) leads to a biased variance estimate. Dividing by $M - 1$ leads to a better variance estimate (*e.g.* [17, 85, 119]). Therefore, for uniformity, the denominator of eq. (2.38) is also set to $M - 1$.

In this work, committees of networks are used for the ML and approximate Bayesian approaches. The reason is two-fold. First, it is unlikely that a single network can yield near-optimal results for a complex task such as curl estimation. Additionally, using committees for the non-resampling approaches allows a fair comparison with the bootstrap technique to be conducted. In chapter 4, the confidence estimation performance of network committees is compared to that of individual networks.

2.6 Summary

This chapter has introduced the basic terminology and methods used in the rest of this thesis.

First, regression estimation using neural networks was briefly introduced. Then, the sources

of uncertainty in neural network predictions were presented. Uncertainty is present in MLP predictions mainly due to

1. the training data, which are typically noisy and sparse, and
2. inefficiencies of the training algorithm and neural network model.

Existing methods for confidence estimation were subsequently presented. The methods can be divided into the following categories:

- The exact and approximate delta methods for model uncertainty variance estimation. These method are adapted from standard non-linear regression theory.
- Methods based on maximum likelihood (ML). These methods can be used to train the network and obtain an estimate of data noise variance. This estimate will be biased and variance may be underestimated.
- Methods based on Bayesian statistics. Only approximate Bayesian methods are considered in the rest of the thesis. Exact methods require Monte Carlo integration over weight space and are impractical for use in the curl estimation problem. The approximate Bayesian approach is feasible and yields an unbiased data noise variance estimate.
- Bootstrap methods. These use a committee of networks to estimate regression and model uncertainty variance. Predictions on the out-of-sample sets can be used to obtain an unbiased data noise variance estimate.

The delta method can be used to obtain an estimate of model uncertainty variance for the ML and approximate Bayesian approaches.

In a real application, the uncertainty estimate must include both uncertainty sources. However, opting for a σ_m^2 estimation approach often conditions the σ_ν^2 estimation approach to be used and vice versa. Moreover, choice of a specific confidence estimation approach may be equivalent to a choice of a regression estimation approach. The available regression and confidence estimation candidate methods to be compared in the next chapters are the following:

- Maximum likelihood for regression and data noise variance estimation. Model uncertainty can be estimated using the approximate or exact delta method.

- Approximate Bayesian method for regression and data noise variance estimation. Model uncertainty can be estimated using the approximate or exact delta method.
- Bootstrap method for regression and model uncertainty estimation. Data noise variance can be estimated by training an additional network on the adapted residuals (or by applying the maximum likelihood formula in the case of constant σ_ν^2) after training is completed.

Chapter 3
Novelty detection for confidence estimation

3.1 Introduction

As mentioned in chapter 2, output reliability depends on the training data density. A network making a prediction for a new, previously unseen input pattern may be *interpolating* when the pattern originates from an input space region which is represented in the training data or *extrapolating* when the input comes from an unrepresented region. In the latter case, the network prediction is not based on information contained in the training data but on *prior* knowledge (*i.e.* on the assumption that the regression function is smooth). Input patterns belonging to regions not represented in the training data are called *novel* inputs. Predictions for novel inputs are unreliable and should be assigned very low confidence.

Novel inputs can be identified by a *novelty detection* scheme. Novelty detection can be viewed as a classification scheme in which each input pattern is classified as belonging to the class of the training data (*i.e.* the "normal" data class) or to a class of novel data. Novelty detection schemes have been used for motor fault prediction [10, 120], medical diagnosis [3] or for deriving a confidence estimate [18].

Novelty detection can be performed by forming a representation of the normal data class and then checking whether new patterns belong to this class or not. This is typically achieved by estimating the unconditional *probability density function* (PDF) of the training (input) patterns [18]. The PDF can be viewed as a measure of the probability that a pattern belongs to a particular data set. Because a novel input bears very little resemblance to the training (normal) data, its associated PDF value will typically be very small. Alternatively, an *auto-associative network* [45, 121] can be used to form a representation of the normal data class. An auto-associative network learns to reproduce input patterns in its output. Therefore, such a network, trained on the set of normal data, will yield very high reproduction error for a novel, dissimilar, input. Thus, novel inputs can be identified by thresholding the PDF or network error. Auto-associative

networks have been used for novelty detection applications [10] although not as confidence estimation tools. In this chapter, an example of novelty detection is presented using the PDF of the data.

In principle, the model uncertainty variance estimate is sensitive to the data PDF. For example, confidence intervals obtained using the approximate Bayesian method and the delta estimator are inversely proportional to the input data density in the neighbourhood of the training data [122]. However, a network may be presented with input patterns that significantly differ from the training data, *i.e.* novel inputs. In section 3.3 it is shown that, for such inputs, model uncertainty variance estimation may fail and the estimated variance may not reflect the true uncertainty level. For this reason, it is necessary, in practice, to gate the prediction system with a novelty detection component (see fig. 3.1).

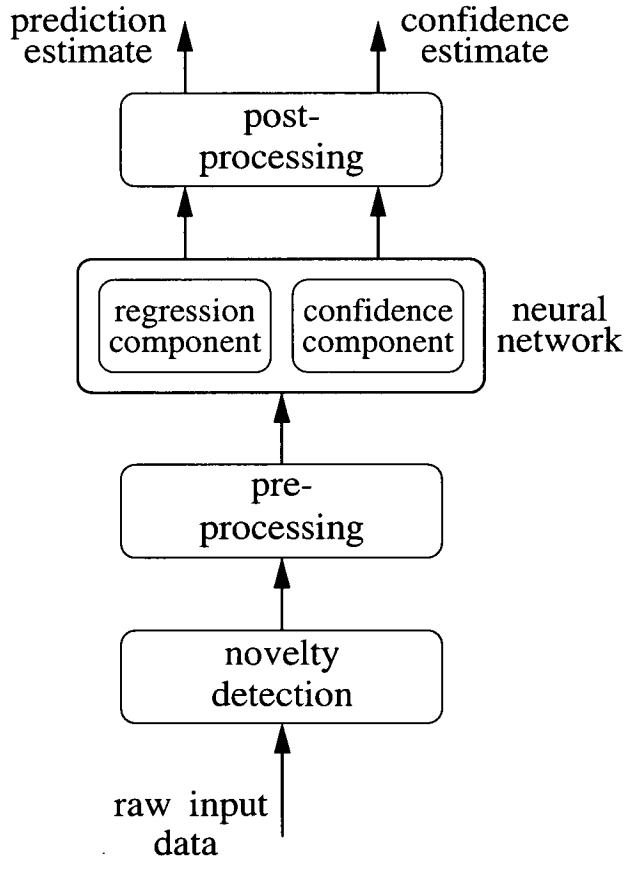


Figure 3.1: *Schematic illustration of a neural prediction system showing the novelty detection, prediction and confidence estimation components as well as the data pre-processing and post-processing stages.*

As mentioned in chapter 2, the training data must be pre-processed before being presented to

the network. If data pre-processing is not information preserving (*e.g.* due to dimensionality reduction), novelty detection must use the raw (*i.e.* unprocessed) input data (see fig. 3.1) to avoid misinterpreting a novel pattern as similar to the training data due to information loss [18].

As far as curl estimation is concerned, two data sets have been used to train the model. Data pre-processing was not information preserving for the first curl set which was used to train the earlier version of the model. In that set, eight of the eleven principal components were retained. Data pre-processing for the current curl model is information preserving since all the principal components are used to train the model. However, in any case, the operator of the curl estimator, used for decision-support, may vary an input parameter manually to investigate its effect on curl. Since the curl data are sparse, the system may well be presented with inputs that differ significantly from the training data. Such inputs may not be identified by the delta or bootstrap estimates of model uncertainty, but can be rejected using a novelty detection scheme.

This thesis focuses on methods for estimating data noise and model uncertainty variance. Therefore, the results presented in the next chapters were obtained using test sets of the same distribution as the training data, *i.e.* no novel input. The aim of this chapter is to demonstrate that novelty detection is necessary in principle for a real system implementation. The rest of the chapter is structured as follows:

Section 3.2 presents Parzen windows, a standard method for estimating the PDF of a data set.

Section 3.3 shows that, in general, the model uncertainty variance estimate may fail to indicate novel inputs. Instead novel inputs can be identified using a novelty detection scheme based on PDF estimation.

3.2 Parzen windows for PDF estimation

Many methods for estimating density functions for novelty detection are available [120, 123–125]. One of the most simple but reasonably effective methods is the *Parzen windows* estimator [126, 127]. The Parzen estimator uses one kernel for each training pattern. If Gaussian kernels are used, the Parzen estimate of the PDF for a previously unseen pattern $\mathbf{x}$ is given by:

$$\hat{p}(\mathbf{x}) = \frac{1}{N(2\pi)^{M/2}s^M} \sum_{i=1}^{N} \exp\left[-\frac{|\mathbf{x} - \mathbf{x}_i|^2}{2s^2}\right] \tag{3.1}$$

where the sum is over input patterns in the training set and M the dimensionality of $\mathbf{x}$. The parameter s controls the degree of smoothness of $\hat{p}(\mathbf{x})$ and can be computed using cross-validation.

The disadvantages of this technique are that the amount of computations increases with the number of available data (since there is one kernel for each training pattern) and that the value of the smoothing parameter s must be chosen manually. In practice, this can be done using cross-validation. Here, Parzen windows are only used for demonstrating purposes, therefore the value of s was heuristically set to the average distance of each training point from its 10 closest neighbours. Furthermore, the novelty detection threshold must be chosen manually, *e.g.* using cross-validation. There exist more advanced methods in which the novelty detection threshold is set automatically during the process of PDF learning [125].

The inverse PDF can be viewed as a measure of model uncertainty variance [18]. However, the inverse PDF is independent of the model at hand and does not scale well. In other words, it is not clear if one can add up the inverse PDF with the data noise variance estimate to obtain the total variance estimate. Therefore, even if novelty detection precedes and "gates" the regression estimation, it is useful to obtain a measure of model uncertainty variance using the delta or bootstrap methods.

3.3 Comparison of novelty detection to model uncertainty estimation

The necessity for novelty detection is demonstrated using the TR_8 curl data set described in appendix B. For these experiments, a demonstrating test set is constructed by varying a single input parameter while holding the rest constant and equal to the input values of a particular test pattern. For each input pattern thus created, the model uncertainty, noise variance and inverse PDF estimates are computed.

Figure 3.2 shows the results for adjusting two different input parameters, x_1 and x_2. Graphs (a) and (b) in fig. 3.2 show the data noise variance estimates (computed using the ML augmented network and bootstrap methods) and the inverse PDF estimate for varying inputs x_1 and x_2 respectively. It is evident that the data noise variance estimates are only valid in the neighbourhood of the training data. The data noise variance estimate does not account for the density of the data.

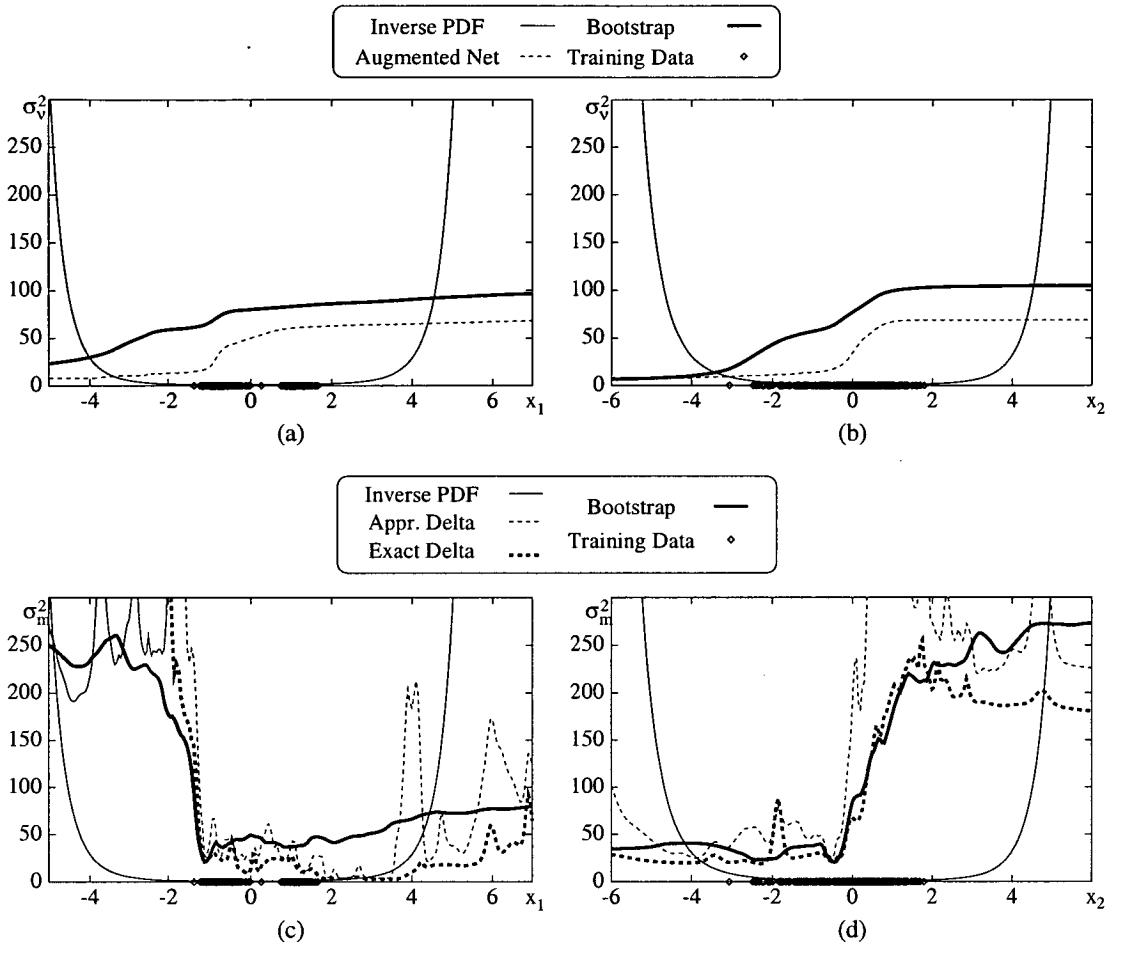


Figure 3.2: *The data noise - (a) and (b) - and model uncertainty variance - (c) and (d) - estimates for varying inputs x_1 and x_2 of the curl set. The rest of the input parameters are kept constant and equal to the values of a test pattern.*

In principle, the model uncertainty variance estimate should account for low data density by rapidly increasing beyond the training data region. Graphs (c) and (d) in fig. 3.2 show three model uncertainty variance estimates and the inverse PDF estimate for varying inputs x_1 and x_2 respectively. Model uncertainty variance was computed using the approximate delta, exact delta and bootstrap method. Occasionally, all three σ_m^2 estimates fail to indicate low density, *i.e.* increase rapidly beyond the region represented in the data. In contrast, the inverse PDF always rises rapidly beyond the high density region.

Therefore, it is necessary to identify low density regions using the PDF of the training data. By thresholding the inverse PDF novel inputs can be identified and rejected. For example, for the case of fig. 3.2, if the threshold value is set to 50, it is evident that the model uncertainty

variance estimates often fail to identify novel inputs. In contrast, the inverse PDF estimate rises rapidly above fifty in the regions of very low training data density. In practice, cross-validation can be used to determine the novelty detection threshold, although more principled techniques are also available [125].

As shown in chapter two, increasing the number of hidden units has the effect of making the model uncertainty estimate more sensitive to low PDF regions. However, the number of hidden units is set so as to achieve optimum generalisation performance. Adding more units unnecessarily may lead to overfitting. Therefore, it is preferable to tackle the problem of identifying novel inputs by thresholding the PDF.

3.4 Summary

This chapter presented novelty detection based on estimating the training data PDF. Novel inputs should be associated with very low confidence because they originate from regions for which the model is not valid. It was shown that the model uncertainty variance estimate may not always identify such inputs. Therefore, it is necessary to precede confidence estimation by a novelty detection scheme that identifies and rejects novel inputs by thresholding the training data PDF.

Chapter 4
Methods for assessing confidence estimation performance

4.1 Introduction

In chapter 2 the candidate approaches to confidence estimation for neural networks were presented. This chapter describes methods for evaluating CM performance and comparing alternative CMs. The issue of evaluating and comparing CM techniques for neural networks has received little systematic study. The first part of this chapter presents previous comparison studies as well as available methods for assessing CM performance quality. Most previously reported comparison studies treat data noise variance as a constant. In this thesis the more general assumption of input-dependent data noise variance is also considered. No comparison study of neural network CMs has ever thoroughly considered the input-dependent noise variance case. Consequently, metrics commonly used for CM performance evaluation have not been tested under conditions of input-dependent noise variance. The second part of this chapter presents a series of comparison studies employing a commonly used CM evaluation technique and considering both constant and input-dependent noise variance.

The chapter is structured as follows:

Section 4.2 gives an overview of previous CM comparison studies.

Section 4.3 describes the *confidence* or *prediction interval coverage probability* (CI or PI CP) metric for CM performance evaluation.

Section 4.4 describes the artificial and real data sets used for evaluating the methods as well as the experimental setup.

Section 4.5 presents simulation results of comparison studies using the CI CP and PI CP metrics.

4.2 A literature survey of CM comparison studies

There exist very few comparison studies of confidence estimation techniques relevant to neural networks.

Donaldson *et al.* [15] present a comparison study of methods for construction of confidence intervals for the parameters of non-linear least squares models. This study is relevant to neural networks because techniques based on *linearization* can be adapted for neural networks [16, 33]. As mentioned in section 2.4.1, these techniques include the exact and approximate delta estimator and the sandwich estimator. In a neural network context, the same techniques are used to obtain estimates of the uncertainty associated with the network output.

The methods were tested by constructing the confidence interval and checking whether the interval contained the true parameter value. The percentage of true values contained in the interval (*i.e.* observed coverage) should be close to the *nominal coverage probability* of the interval. The nominal value of the coverage probability (CP) depends upon the output distribution and the width of the intervals. A CM is considered optimum if it consistently yields observed CP close to the nominal value.

Linearization methods are based on the assumption that the non-linear function can be approximated by an affine or linear approximation at the solution. In general, this assumption does not hold exactly for non-linear models such as neural networks. Therefore, the confidence intervals obtained by the delta method are only approximate, *i.e.* the observed coverage may not equal the theoretical coverage of the interval. The approximate delta method was found to outperform the other variants. The reason is that evaluation of the exact Hessian is often unstable for non-linear models. This study considered only constant data noise variance.

The observed coverage probability can also be used to obtain a merit of quality for CMs accompanying the predictions of a MLP [16, 20]. In this case, the true parameter values are the true regression values (for confidence intervals) or the targets (for prediction intervals). For example, in fig. 4.1 the network output and prediction intervals are shown for a one-dimensional data set. Two profiles - (b) and (c) - show the cases of target lying out of the interval (fig. 4.1 (b)) and target contained in the interval (fig. 4.1 (c)). As in [15], the observed coverage can be estimated as the percentage of targets that lie within the interval, using the available test data.

Tibshirani [73] compares three model uncertainty estimation techniques for neural networks,

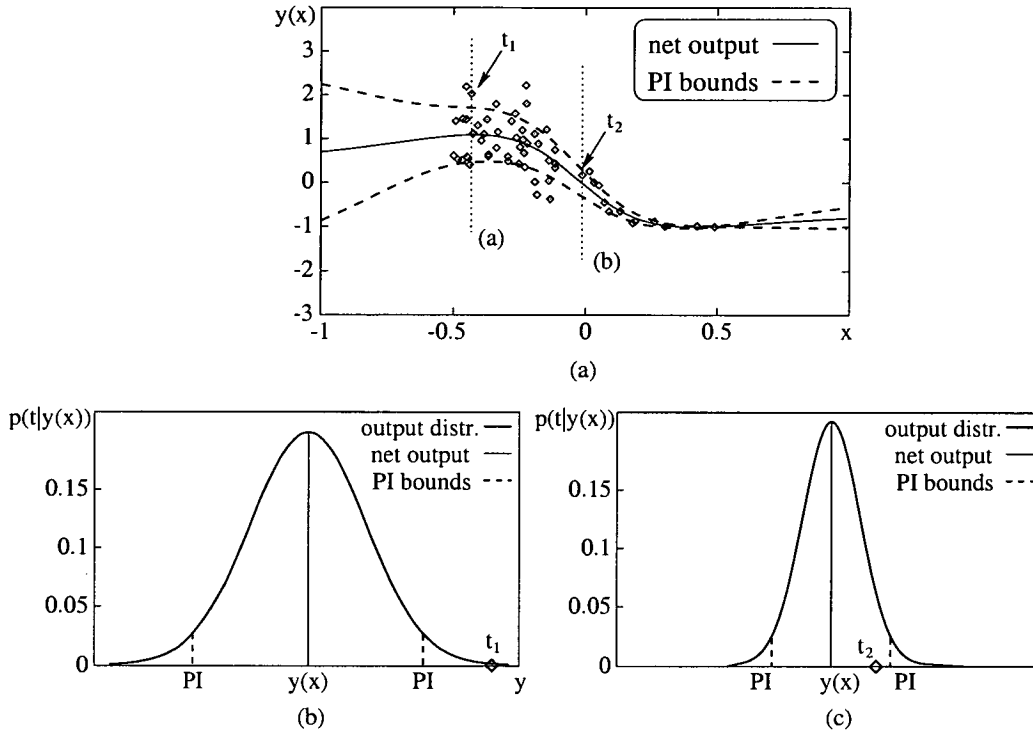


Figure 4.1: *Graph (a) shows the network output, prediction intervals, and test target values for an one-dimensional regression task. In profile (b) the true value of the parameter (in this case target t_1) is not contained in the interval while, in profile (c) target t_2 is contained in the interval.*

namely the approximate and exact delta estimators, the sandwich estimator, (using both the exact and the approximate Hessian) and the bootstrap method using pairs or residual sampling. Performance is evaluated by comparing the estimated model uncertainty variance with the "true" value. An estimate of the "true" model uncertainty variance is obtained by performing a Monte Carlo simulation over different realizations of the same artificial data (see fig. 4.2). Then, the median of the estimated variance over all test patterns is compared with the median of the true values ms.

The bootstrap pairs sampling method was found to yield estimates closer to the real value than the rest of the methods. However, the result for the non-simulation methods (delta and sandwich) was computed using a single network and the networks were trained using a simplistic - constant learning rate - optimisation algorithm that is more susceptible to the choice of initial weight values than the more sophisticated techniques used in this work. Thus, the non-simulation methods missed the variability due to the choice of initial random weights and the variance was underestimated. As in [15], only constant σ_ν^2 was considered.

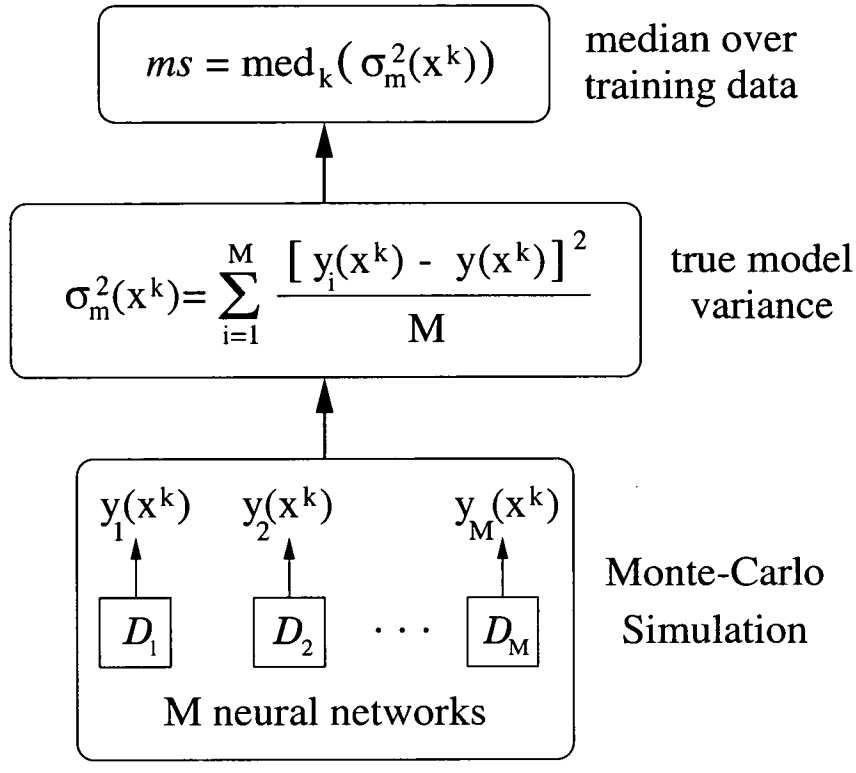


Figure 4.2: *Estimation of the median ms of the true model uncertainty variance. The true model uncertainty variance $\sigma_m^2(\mathbf{x}^k)$ for data point $\mathbf{x}^k$ is estimated using M different data set realizations $\mathcal{D}_1$, $\mathcal{D}_2$,..., $\mathcal{D}_M$ created using the same (artificial) data generating mechanism. $y(\mathbf{x}^k)$ is the average prediction over the M simulations given as $y(\mathbf{x}^k) = \sum_{i=1}^{M} y_i(\mathbf{x}^k)/M$.*

Finally, Lowe *et al.* [68] present a comparison study of three approaches to confidence interval estimation. An one-dimensional toy problem and a real industrial problem (engine calibration) are employed to investigate the performance of *Gaussian Processes*, the *approximate Bayesian approach* with constant σ_ν^2 and *predictive error bars*. Predictive error bars are the error estimates obtained by an additional network which is trained with the same inputs and the squared errors of the original network as targets. As mentioned in chapter 2, this estimate is essentially the same as the maximum likelihood estimate of σ_ν^2 without the assumption that the prediction distribution is Gaussian. Thus, the error bar cannot be interpreted as variance of the prediction distribution. It was found that predictive error bars were more appropriate for that particular application because the estimate of uncertainty followed the variation in the data noise.

4.2.1 Criticism of CM comparison studies

Both [15] and [73] dealt only with model uncertainty estimation. However, as mentioned in chapter 2, a neural network confidence estimation system should compute an estimate of data noise variance as well. This is because in a real application, the network is predicting noisy targets rather than the non-noisy regression function. In both those studies, data noise variance was estimated (when required for computing the σ_m^2 estimate) as a constant using the maximum likelihood estimate of eq. (2.19) [73], repeated here for convenience

$$\hat{\sigma}_\nu^2 = \frac{1}{N}\sum_{n=1}^{N}[t^n - y(\mathbf{x}^n; \hat{\mathbf{w}})]^2 \tag{4.1}$$

or the Bayesian estimate for the case of non-linear models [15]

$$\hat{\sigma}_\nu^2 = \frac{1}{N-P}\sum_{n=1}^{N}[t^n - y(\mathbf{x}^n; \hat{\mathbf{w}})]^2 \tag{4.2}$$

where P is the number of model parameters. Thus, alternative data noise variance estimation techniques were not compared and, more importantly, the case of input-dependent data noise variance was not considered.

As far as [68] is concerned, model uncertainty is not treated as an independent source from data noise and the adopted CM approach only deals with data noise variance. This approach works for the particular application studied there, probably because the problem is constrained so that novel data can never be encountered and the training data density is relatively high. However, even if this is the case, it is still worthwhile to include model uncertainty estimates because the PDF may not be uniform over the input space. Therefore, the conclusions of [68] do not generalise to problems such as curl estimation for which the available data are limited.

4.2.2 Choosing the evaluation tools

There exist two alternative methods of CM performance evaluation:

1. Confidence measure performance can be evaluated by computing the observed coverage probability of the confidence or prediction intervals and comparing it with the theoretical value [15, 16, 20].

2. Alternatively, the real model uncertainty variance can be estimated by performing a Monte-Carlo simulation over different realizations of the same artificial data [73]. Then the estimated values can be compared with the real ones. Likewise, if data noise variance is also estimated, the estimated values can be compared with the real ones (which are known for artificially generated data).

In this chapter, the confidence or prediction interval coverage probability (CI CP or PI CP) is used for CM performance evaluation. The comparison approach used in [73] was not favoured for the following reasons:

- It can only be used with artificial data.
- It is unclear how many times each experiment must be repeated to obtain a good estimate of the real model uncertainty variance.
- In practice, the estimate of the real model uncertainty variance may be different for different networks trained under different training regimes (*e.g.* ML and Bayesian training).
- Finally, considering that each experiment must be repeated an adequate number of times to minimise variance error in the results and that the computing resources and time available for the completion of this project were limited, this approach is computationally impractical compared to PI CP.

4.2.3 An improved CM comparison study

To sum up, the comparison study presented in this chapter is novel and superior to previous studies for the following reasons:

- It is the first study of ANN confidence measures where both data noise and model uncertainty are explicitly considered. There are previous works that also consider both sources but they do not offer a complete comparison with other existing methods (*e.g.* [19] considers both uncertainty sources but the proposed approach is compared only to Nix and Weigend's [27] incomplete approach).
- There is no complete comparison study of these techniques using input-dependent data noise variance. This is a serious limitation as in many real problems the noise may not have constant variance.

- The approximate Bayesian approach with input-dependent data noise variance has not been thoroughly applied and tested using neural networks. This approach was originally developed for *generalised linear models* and holds only approximately for neural networks [17].

The next section describes the use of coverage probability for CM performance evaluation in more detail.

4.3 The interval coverage probability metric

Confidence and prediction intervals are error bars within which the true value lies with a known probability, called the coverage probability (CP). The probability that the target lies within the interval has a theoretical value given by the interval characteristics. CM performance can be assessed by computing the observed CP using some test data and comparing it with the theoretical value. Ideally, a CM should consistently yield observed coverage close to the nominal value.

4.3.1 Confidence and prediction intervals

Confidence intervals (CIs) quantify how well the network models the actual regression task, *i.e.* how well the network function $y(\mathbf{x})$ estimates the true function $f(\mathbf{x})$. In other words, CIs refer to the probability distribution $p(f(\mathbf{x})|y(\mathbf{x}))$ that the true regression is $f(\mathbf{x})$ given the estimate $y(\mathbf{x})$ or equivalently, to the distribution of the quantity $f(\mathbf{x}) - y(\mathbf{x})$ [19]. In the approach adopted here, this distribution is Gaussian with zero mean and variance which is estimated as $\hat{\sigma}_m^2(\mathbf{x})$, the model uncertainty variance estimate.

Prediction intervals (PIs) refer to the model's ability to predict targets. Therefore, PIs refer to the probability distribution $p(t|y(\mathbf{x}))$ that the true target is t given the network prediction $y(\mathbf{x})$. Equivalently, PIs deal with the distribution of $t - y(\mathbf{x})$. Using eq. (2.1), it follows that

$$t - y(\mathbf{x}) = [f(\mathbf{x}) - y(\mathbf{x})] + e \tag{4.3}$$

where e is the data noise component. Thus, the prediction interval includes the confidence interval. Because e and $f(\mathbf{x}) - y(\mathbf{x})$ are statistically independent [19, 23] the variance of $t - y(\mathbf{x})$

is given by

$$<[t - y(\mathbf{x})]^2>= <[f(\mathbf{x}) - y(\mathbf{x})]^2>+<e^2>= \sigma_m^2(\mathbf{x}) + \sigma_\nu^2(\mathbf{x}) = \sigma_{\text{TOTAL}}^2(\mathbf{x}) \tag{4.4}$$

Therefore, prediction intervals are computed using the total standard deviation

$$\sigma_{\text{TOTAL}}(\mathbf{x}) = \sqrt{\sigma_\nu^2(\mathbf{x}) + \sigma_m^2(\mathbf{x})} \tag{4.5}$$

In a real task the network is predicting target values. In this chapter, therefore, CM performance is investigated using the PI in preference to CI coverage probability. However, results using the CI CP are also presented in order to perform an explicit comparison of model uncertainty estimation methods.

4.3.2 Nominal Coverage probability

Coverage probability (CP) is the probability that the true value of the estimated quantity lies within the interval. When confidence intervals are considered the CP refers to the probability that the true regression value lies within the interval. If prediction intervals are used, the CP is the probability that the target lies within the interval. The *nominal* or *theoretical* coverage of intervals of a given size is determined by the form of the probability distribution of the network output.

In this study Gaussian distributions are used for both the data noise and network output distributions. Therefore, for an interval of k standard deviations width $(y(\mathbf{x}) \pm k\sigma(\mathbf{x}))$, the nominal coverage probability can be computed by integrating the Gaussian curve (see fig. 4.3) between the interval bounds

$$P(t \in \text{PI})_{\text{nominal}} = \int_{y(\mathbf{x})-k\sigma(\mathbf{x})}^{y(\mathbf{x})+k\sigma(\mathbf{x})} \frac{1}{\sqrt{2\pi}\sigma(\mathbf{x})} \exp\left(-\frac{[k - y(\mathbf{x})]^2}{2\sigma^2(\mathbf{x})}\right) dk \tag{4.6}$$

This integral can be computed using numerical techniques [50]. For the two commonly used interval widths of one and two standard deviations ($y \pm \sigma$ and $y \pm 2\sigma$) the nominal coverage is equal to 68% and 95% respectively. In the results of this chapter all intervals have width of two standard deviations ($y \pm 2\sigma$) so that the nominal CP is 95%.

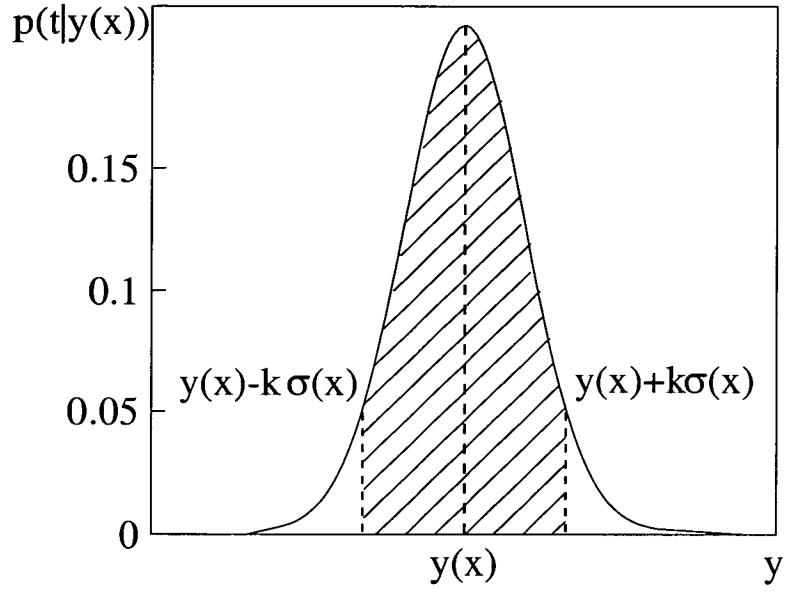


Figure 4.3: *Estimation of the nominal coverage probability of a Gaussian prediction interval with bounds* $(y(x)-k\sigma(x), y(x)+k\sigma(x))$. *The nominal CP equals the area below the Gaussian curve and delimited by the interval bounds.*

4.3.3 Observed coverage probability

The performance of a confidence measure can be assessed in practice by computing the observed CP and comparing it with the nominal value [15, 16, 20]. A common method for computing the observed coverage probability of a given model and CM is to calculate the percentage of test patterns for which the target lies within the corresponding PI about the network prediction [15, 16]. The "percentage of covered targets" is effectively an estimate of the mean CP value, *i.e.* averaged over the entire input space. This method is straightforward to use for artificial as well as real data and can be employed to compute the observed coverage of both confidence and prediction intervals.

If artificial data are used, instead of computing the percentage of targets covered by the interval, the prediction interval coverage probability can be estimated by calculating the probability that the target lies within the interval at every test point, and then averaging over all test points [20]. At each test point, the probability that the target lies within the interval is estimated by calculating the area of the true target distribution curve that falls within the PI (see fig. 4.4). This method can be expected to yield a more robust observed coverage estimate than the former approach which merely checks whether the target $f(\mathrm{x}) + e$ happens to lie within the interval for a specific value of e. Additionally, this method can be used to compute the standard deviation

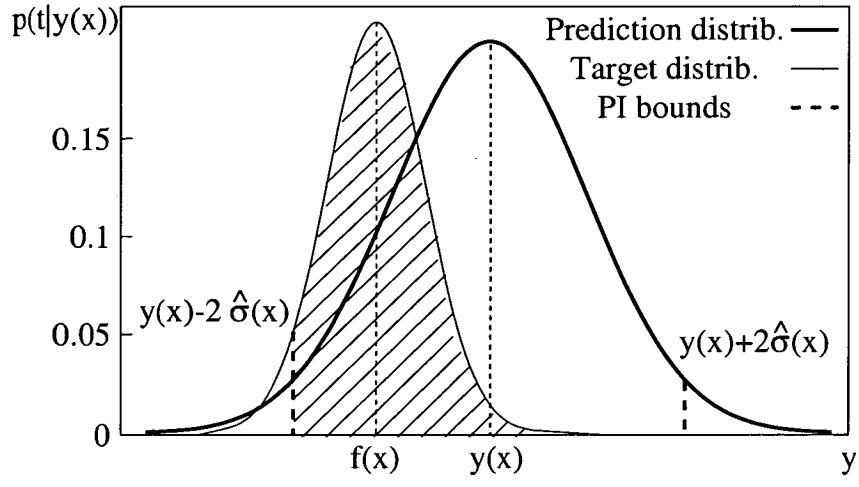


Figure 4.4: *The probability that the target lies within the prediction interval of two standard deviations width for a particular test pattern is given by the (shaded) area below the curve of the true target distribution* $N(f(\mathbf{x}), \dot{\sigma}_\nu^2(\mathbf{x}))$ *and delimited by the interval bounds* $(y(\mathbf{x}) - 2\hat{\sigma}(\mathbf{x}), y(\mathbf{x}) + 2\hat{\sigma}(\mathbf{x}))$.

of the CP over the test patterns, because a measure of coverage is available at each test point.

The observed coverage probability at a particular test point can be estimated as follows. As shown in fig. 4.4, the probability $P(t \in \text{PI})$ that the target lies within the 2σ PI is given by the area below the curve of the true target distribution $N(f(\mathbf{x}), \dot{\sigma}_\nu^2(\mathbf{x}))$ and between the limits $(y(\mathbf{x}) - 2\hat{\sigma}(\mathbf{x}), y(\mathbf{x}) + 2\hat{\sigma}(\mathbf{x}))$. Using

$$\frac{t - f(\mathbf{x})}{\sqrt{2}\dot{\sigma}_\nu(\mathbf{x})} = k \Leftrightarrow t = \sqrt{2}\dot{\sigma}_\nu(\mathbf{x})k + f(\mathbf{x}) \tag{4.7}$$

it follows that

$$\begin{aligned} P(t \in \text{PI}) &= \frac{1}{\sqrt{2\pi\dot{\sigma}_\nu(\mathbf{x})^2}} \int_{y(\mathbf{x})-2\hat{\sigma}(\mathbf{x})}^{y(\mathbf{x})+2\hat{\sigma}(\mathbf{x})} \exp\left\{-\frac{(t - f(\mathbf{x}))^2}{2\dot{\sigma}_\nu(\mathbf{x})^2}\right\} dt = \\ &= \frac{\sqrt{2}\dot{\sigma}_\nu(\mathbf{x})}{\sqrt{2\pi}\dot{\sigma}_\nu(\mathbf{x})} \int_A^0 \exp(-k^2)dk + \frac{\sqrt{2}\dot{\sigma}_\nu(\mathbf{x})}{\sqrt{2\pi}\dot{\sigma}_\nu(\mathbf{x})} \int_0^B \exp(-k^2)dk = \\ &= 0.5 \cdot \frac{2}{\sqrt{\pi}} \int_0^B \exp(-k^2)dk - 0.5 \cdot \frac{2}{\sqrt{\pi}} \int_0^A \exp(-k^2)dk \end{aligned} \tag{4.8}$$

where $\dot{\sigma}_\nu(\mathbf{x})$ is the real data noise standard deviation. For clarity, the integral limits have been

renamed as A and B which are given by

$$A = \frac{y(\mathbf{x}) - 2\hat{\sigma}(\mathbf{x}) - f(\mathbf{x})}{\sqrt{2}\dot{\sigma}_\nu(\mathbf{x})} \tag{4.9}$$

$$B = \frac{y(\mathbf{x}) + 2\hat{\sigma}(\mathbf{x}) - f(\mathbf{x})}{\sqrt{2}\dot{\sigma}_\nu(\mathbf{x})} \tag{4.10}$$

Therefore, using the definition of the *error function*[1] $\mathrm{erf}(x)$ [50, 128]

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-k^2) dk \tag{4.11}$$

it follows that the probability that the target lies within the $(y(\mathbf{x}) \pm 2\hat{\sigma}(\mathbf{x}))$ prediction interval is given by

$$P(t \in \mathrm{PI}) = 0.5 \cdot \mathrm{erf}\left(\frac{y(\mathbf{x}) + 2\hat{\sigma}(\mathbf{x}) - f(\mathbf{x})}{\sqrt{2}\dot{\sigma}_\nu(\mathbf{x})}\right) - 0.5 \cdot \mathrm{erf}\left(\frac{y(\mathbf{x}) - 2\hat{\sigma}(\mathbf{x}) - f(\mathbf{x})}{\sqrt{2}\dot{\sigma}_\nu(\mathbf{x})}\right) \tag{4.12}$$

where $\hat{\sigma}(\mathbf{x})$ is the estimated total standard deviation. In computer simulations the error function integral can be computed using numerical procedures [50]. The observed CP mean can be computed by estimating the CP for each test point using eq. 4.12 and then obtaining the average. In this thesis, this estimate of the observed CP mean is called the "estimated coverage probability" to distinguish between this approach and the estimate of the observed CP mean as the percentage of covered targets.

The true target distribution is naturally known only for artificial data sets. However, the estimated coverage probability can be computed at each test point for real data sets, if the network prediction is assumed to be a good estimate of the real regression and the model residual a good estimate of the true data noise standard deviation. In this case, eq. 4.12 reduces to

$$P(t \in \mathrm{PI}) = 0.5 \cdot \mathrm{erf}\left(\frac{2\hat{\sigma}(\mathbf{x})}{\sqrt{2}r(\mathbf{x})}\right) - 0.5 \cdot \mathrm{erf}\left(\frac{-2\hat{\sigma}(\mathbf{x})}{\sqrt{2}r(\mathbf{x})}\right) \tag{4.13}$$

where $r(\mathbf{x}) = |y(\mathbf{x}) - t|$ is the model residual. Note that, effectively, this equation checks the quality of the data noise variance estimate. Thus, an estimate of the CP standard deviation can be computed for both artificial and real data.

[1] Not to be confused with the function used to train a network.

4.3.4 Interpretation of the CP mean and standard deviation

As mentioned above, the estimated mean CP is obtained by averaging over all the test points. If the total standard deviation estimate $\hat{\sigma}_{\text{TOTAL}}$ is, on average, a good estimate of the actual uncertainty, the estimated mean CP will have a value close to the nominal. If $\hat{\sigma}_{\text{TOTAL}}$ either underestimates or overestimates the actual uncertainty on average, the estimated mean CP will be smaller or larger than the nominal value. Either discrepancy indicates a sub-optimal confidence measurement technique.

In practice, the observed CP result depends on the size of the interval (*e.g.* one or two standard deviations). This effect is particularly important for small test sets. For example, the test sets of the two curl sets, used in the comparison studies in the end of this chapter, contain 185 and 224 patterns respectively. Therefore, for intervals of 2σ width, it is expected that about 5% of the test patterns, *i.e.* about 9 and 11 patterns respectively, will lie out of the interval. Clearly, this is a small number of patterns and the observed CP result may suffer high variance error. A possible solution would be to use variable interval size and report results for all sizes. Alternatively, a normality test, such as K^2 [129], can be used to check the similarity of the observed and nominal probability distributions. Finally, the *likelihood* for the regression and error models [34] can be used as a measure of their quality. The latter two approaches (which are not investigated here) have the advantage of being independent of the actual interval size. As far as artificial sets are concerned, the problem can be remedied easily because a sufficiently large test set can be constructed. In this work, the test sets of the artificial sets contain 10000 examples and intervals of 2σ width are used for all the reported results.

Clearly, the mean coverage probability is only sensitive to the average quality of the confidence interval over the entire input space. However, the ideal CM should also yield good *local* σ_{TOTAL} estimates. The local quality of the σ_{TOTAL} estimate can be gauged by computing $\hat{\sigma}_{\text{cp}}$, the standard deviation of the estimated coverage probability over the test points. A confidence measure that yields good mean CP but high CP standard deviation over the test points is likely to be producing inaccurate σ_{TOTAL} estimates locally.

The concept of local CM performance is illustrated with a stylised example in fig. 4.5. The prediction intervals of one standard deviation width are shown for an artificial data set of one-dimensional input. Fig. 4.5 (a) shows the prediction intervals computed treating σ_ν^2 as input-dependent and fig. 4.5 (b) using constant σ_ν^2. Here the true data noise variance is input-

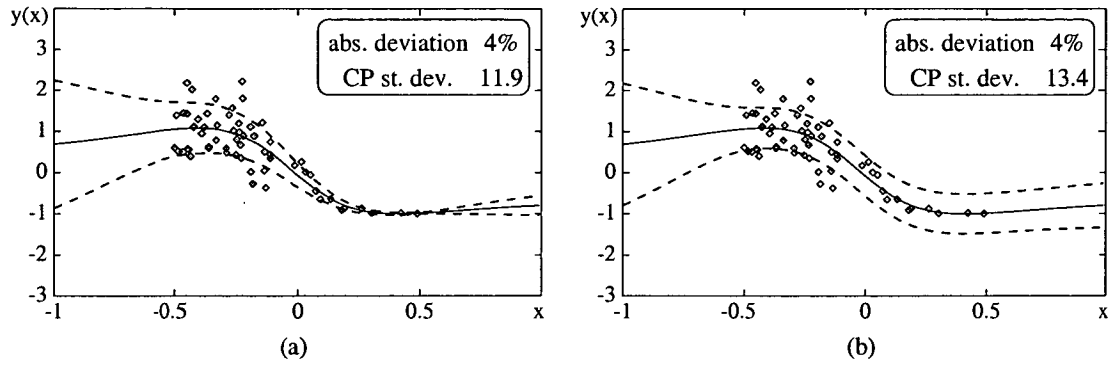


Figure 4.5: *Schematic illustration of the concept of local CM performance. The observed mean coverage for both CMs has similar absolute deviation from the nominal value (68%), however, CM (b) is much worse locally. This is indicated by larger CP standard deviation.*

dependent, so intervals (a) follow the local variations in uncertainty better than (b) and are clearly more realistic. Both CMs exhibit similar observed CP mean (both approximately 4% from the nominal value). This is because locally bad uncertainty estimates (see fig. 4.5 (b)) may compensate for one another resulting in an overall good observed CP estimate. However, CM (b) exhibits a larger CP standard deviation, indicating poorer local performance.

In this stylised example the CP standard deviation correctly indicates suboptimal local performance. In section 4.5, the ability of $\hat{\sigma}_{\text{cp}}$ to gauge local performance is investigated more rigorously, using artificial data of constant and input-dependent noise variance.

4.4 Experimental setup

The alternative confidence estimation techniques have been tested on both artificial and real problems. For artificial data, the true answer is known and arbitrarily large test sets can be constructed. This is impossible for the curl problem, and all real problems, for which data collection is time-consuming and expensive. The probabilistic nature of the comparison tools means that larger rather than smaller test sets are preferable to approximate asymptotic behaviour (*i.e.* behaviour for infinite sets).

4.4.1 Network training approach

All the networks are trained using steepest descent [34] and line search with Brent's algorithm [50, 58]. Weight decay with the same (constant) regularization parameter was used. Using more

sophisticated methods to choose the regularization parameters value (*e.g.* evidence framework [30] or cross-validation) would probably lead to better generalization performance. However, this work is only concerned with comparing a set of confidence estimation techniques on equal footing, rather than determining the best possible model for the available data sets. For this purpose, using a constant value for the regularization parameter suffices. Moreover, validation was not used as it is not used in the approximate Bayesian approach. If it was used in ML training then this method would suffer the disadvantage of not having seen a significant proportion of the data during training. Therefore, all networks were trained for a fixed number of epochs, until a sufficient, similar level of learning was achieved. Details of the training procedure can be found in App. A.

In reality a neural network trained on a finite data set suffers from *bias* and *variance* error [130]. Bias error occurs because the average (over all possible realizations of the data set) function learned by the network may differ from the true function. For example, neural networks have a tendency to over-smooth the input-output mapping. Variance error occurs because the network function may be sensitive to the particular data set at hand. Moreover, variance error may be affected by the training regime, the choice of initial weights, *etc* . To reduce variance error in the results of this study, 300 networks were trained and the reported results are the average over 500 committees formed by choosing networks at random from the pool. Clearly, the results still suffer error due to the finite data set, the particular realisation of the artificial data sets, the choice of the regularization parameter value, *etc* . However, a pragmatic choice had to be made, taking into account the resources and time available.

As mentioned in chapter two, committees of networks were used for the non-simulation approaches (ML and Bayesian). Using committees for the ML and Bayesian methods allows a fair comparison with the bootstrap technique to be conducted. Moreover, it is unlikely that a single network can ever yield near-optimal results for a real, complex task. The committee size was set to 20 networks for all methods. This is a reasonable minimum committee size for practical applications where training and query time must be taken into consideration. For the ML and Bayesian methods the committee prediction and total variance are given by eq. (2.36) and (2.37) respectively.

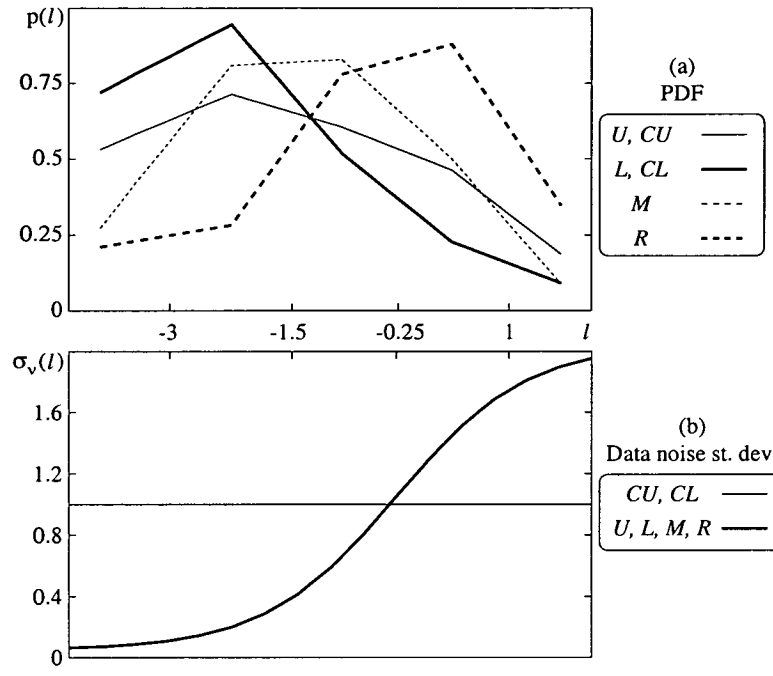


Figure 4.6: *The normalised average PDF (graph (a)) and the true data noise standard deviation (graph (b)) of the artificial data sets across their first principal component* $l(\mathbf{x})$*:* $l(\mathbf{x}) = x_1 + x_2 - 2x_3 - 5x_4 + 2x_5$.

4.4.2 The artificial data sets

The artificial tasks are variants of a benchmark problem proposed in [131]. The input is five-dimensional and the targets are given by

$$t = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + e \tag{4.14}$$

Each x_i is randomly sampled from $(0, 1)$. The errors e have Gaussian distribution with zero mean and constant or input-dependent standard deviation σ_ν.

Six artificial data sets are used to investigate performance under different input data density configurations (see also fig. 4.6):

- sets *U* and *CU* have uniform PDF, *i.e.* x_i is uniformly distributed in $(0, 1)$ for each $i = 1, ..., 5$;
- set *L* and *CL* have higher data density in the region of the input space where data noise variance is low;

- set M has higher data density in the average data noise variance region, and
- set R has higher data density in the high data noise variance region.

In other words, set L for example, contains less training data originating from the input space region of high σ_ν values and more data from the region of low σ_ν values. This experimental setup was chosen to investigate the combined effects of the distribution of training data density and the form of the variance of the data noise. Qazaz [17] shows that the bias in the ML σ_ν^2 estimate has an effect largely in regions of low data PDF and high data noise. Each training set contained 120 examples and a separate set of 10000 examples was used for testing. The regression and variance hidden layers contained five and one units respectively and the regularization parameter was set to 0.01. It is assumed that the network architecture is optimised. In practice, this assumption may not be valid for real, small data sets. However, in practice, the assumption that the best available architecture is optimum must be adopted when modelling a real complex problem because the actual optimum architecture is not and can not be known (due to the small data sample).

Sets CU and CL have constant $\sigma_\nu(\mathbf{x}) = 1$ whereas sets U, L, M and R have input-dependent σ_ν given by

$$\sigma_\nu(\mathbf{x}) = 2g(l(\mathbf{x})) + 0.05 \tag{4.15}$$

$$l(\mathbf{x}) = x_1 + x_2 - 2x_3 - 5x_4 + 2x_5 \tag{4.16}$$

where $g()$ is the logistic sigmoid function. Thus, the artificial data sets have constant or input-dependent data noise variance and various PDF configurations.

4.4.3 The curl data sets

The real data are the *paper curl* data described in [22]. The data have been collected in the Tullis-Russell paper-making plant. Curl is an important paper quality parameter whose value can only be measured after manufacture.

There are 672 input-output pairs available. Ten parameters of the paper production process are used as inputs to the neural network. There is one categorical variable, *paper grade*, which can take three values and is encoded using two input units as explained in [22]. Therefore, the number of input units increases to eleven. The predictor variables (*i.e.* input parameters) used

to model curl are described in appendix B.

The first version of the curl model was trained using the largest eight principal components as inputs and worked for a particular grade of paper only [22]. Therefore, that data set contained only 554 of the available patterns. The current version of the curl model uses all 11 principal components and all the available data. In this thesis, both curl sets are used. The eight-input set is denoted TR_8 and the eleven-input set TR_{11}. In this work the two data sets have been divided into training and test sets as follows:

- The 554 patterns of set TR_8 are divided into two sets of 369 and 185 examples which are used for training and testing the model respectively.
- The 672 patterns of set TR_{11} are divided into a training set of 448 patterns and a test set of 224 patterns.

Clearly, both available curl data sets are very sparse.

Curl prediction was performed in [22] using a committee of networks, under the assumption of constant data noise variance. Here, the effect of treating σ_ν^2 as a function of the inputs is investigated. The constant variance approach serves as a "baseline" against which the corresponding input-dependent σ_ν^2 approach is compared.

As far as set TR_8 is concerned, it was found that one hidden unit is enough to model σ_ν^2 while nine units were used for modelling the regression. As set TR_{11} is the one used to train the current curl model, the network architecture used in this study is the same as the one used in the curl estimator. The regression is modelled by twelve hidden units and the variance by five. The number of hidden units for modelling curl was chosen after extensive experimentation, taking into account the experience of previous modelling attempts [22]. For both sets, the regularization parameter was set to 0.01 for both the regression and the variance networks[2].

[2]Set TR_{11} was used to train the current version of the curl model using the approximate Bayesian approach with EF and input-dependent data noise variance. Thus, the baseline and augmented approximate Bayesian models for set TR_{11} use variable regularization parameter values as estimated by the EF procedure.

4.5 Results

Results of six comparison studies are presented. The first three comparisons are preliminary. Their purpose essentially is to determine the optimum tools to be used for the final three comparisons which comprise the main body of this chapter's results.

The main comparison studies are as follows. First, CMs that treat data noise variance as constant are compared to each other. Then, a comparison study of all the considered CMs under the assumption of input-dependent data noise variance is presented. Finally, a comparison study of CMs that consider constant data noise to CMs that assume input-dependent noise is performed.

Since the comparison criterion is the difference of the observed CP from the nominal value, the graphs show the absolute difference (observed minus nominal CP) as boxes. Negative differences (*i.e.* cases in which the observed CP value was smaller than the nominal) are shown in red colour boxes, while positive differences (*i.e.* when the observed CP value was higher than the nominal) in black colour boxes. All the intervals have nominal coverage of 95% (*i.e.* the interval width is two standard deviations).

4.5.1 Comparison of the exact to the approximate delta method

In previous studies it was found that the approximate delta method that uses the outer-product approximation to the Hessian performs at least as well as the exact delta method that uses the exact Hessian [15]. Since computation of the approximate Hessian is much faster and generally more stable than computing the exact Hessian it would be preferable to use the approximate in preference to the exact delta in this work.

Therefore, the exact and approximate delta methods are compared using the CI CP and PI CP metrics. The observed confidence interval coverage probability (CI CP) only concerns the model uncertainty estimation component of the confidence measure, *i.e.* the approximate and exact delta methods or the bootstrap technique. Only the observed mean value of the CI CP can be computed as the percentage of the true regression values that lie within the interval. Naturally, the CI CP can be computed only for the artificial data sets for which the true regression is known. Figure 4.7 presents the CI CP results for the data sets used in this thesis. It is evident that, on average, the approximate delta method outperforms the exact. In cases where the approximate Hessian gives worse results the difference in performance is negligible.

Figure 4.8 presents the PI CP results for the the two delta methods and the ML and Bayesian committees. The observed mean PI CP was computed using the estimated CP method in which the CP is first estimated at each test point and then averaged over all test points. The PI CP concerns the total standard deviation estimate that comprises a model uncertainty estimate and a data noise variance estimate. This time results are reported for the curl sets as well. Note that the results for the curl sets should be viewed with caution due to the very small test sets (the TR_8 and TR_{11} test sets contain 185 and 224 patterns respectively). Again the approximate delta method is superior to the exact in most cases. Therefore, in the rest of this work, model uncertainty variance for ML and Bayesian networks is computed using the approximate delta method.

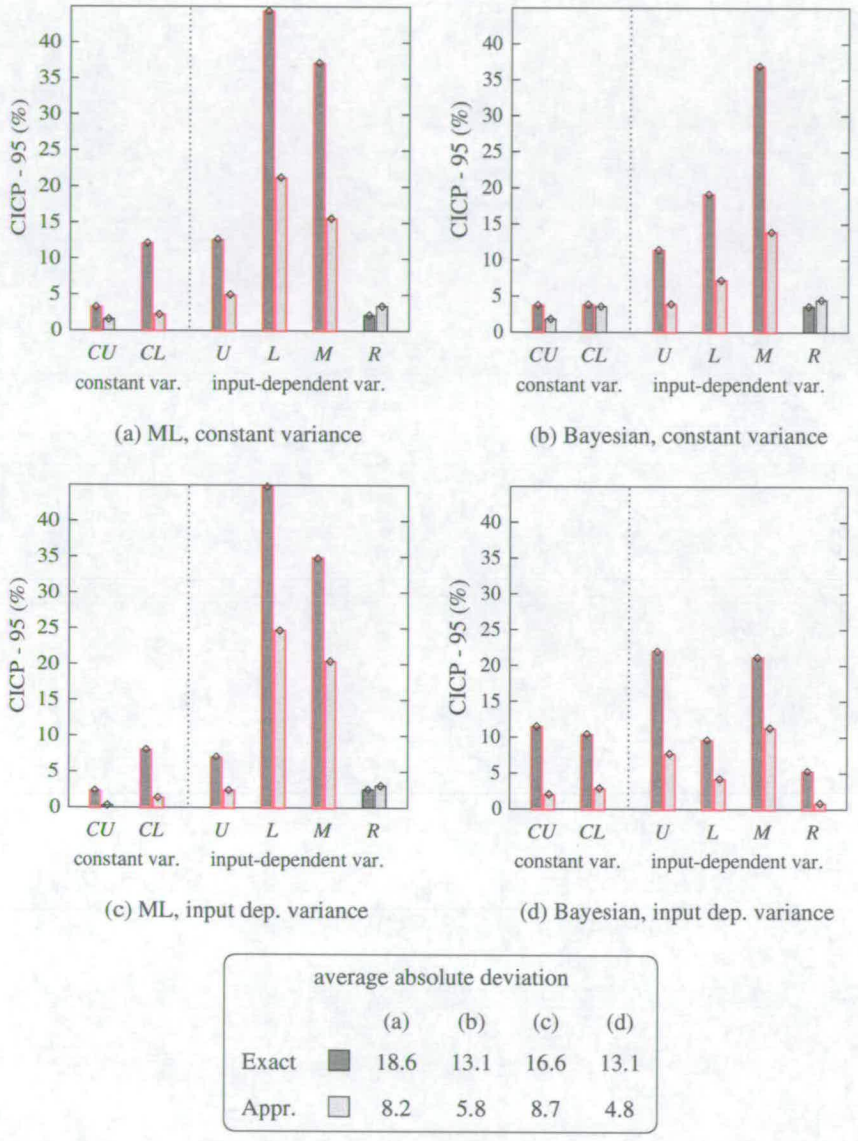
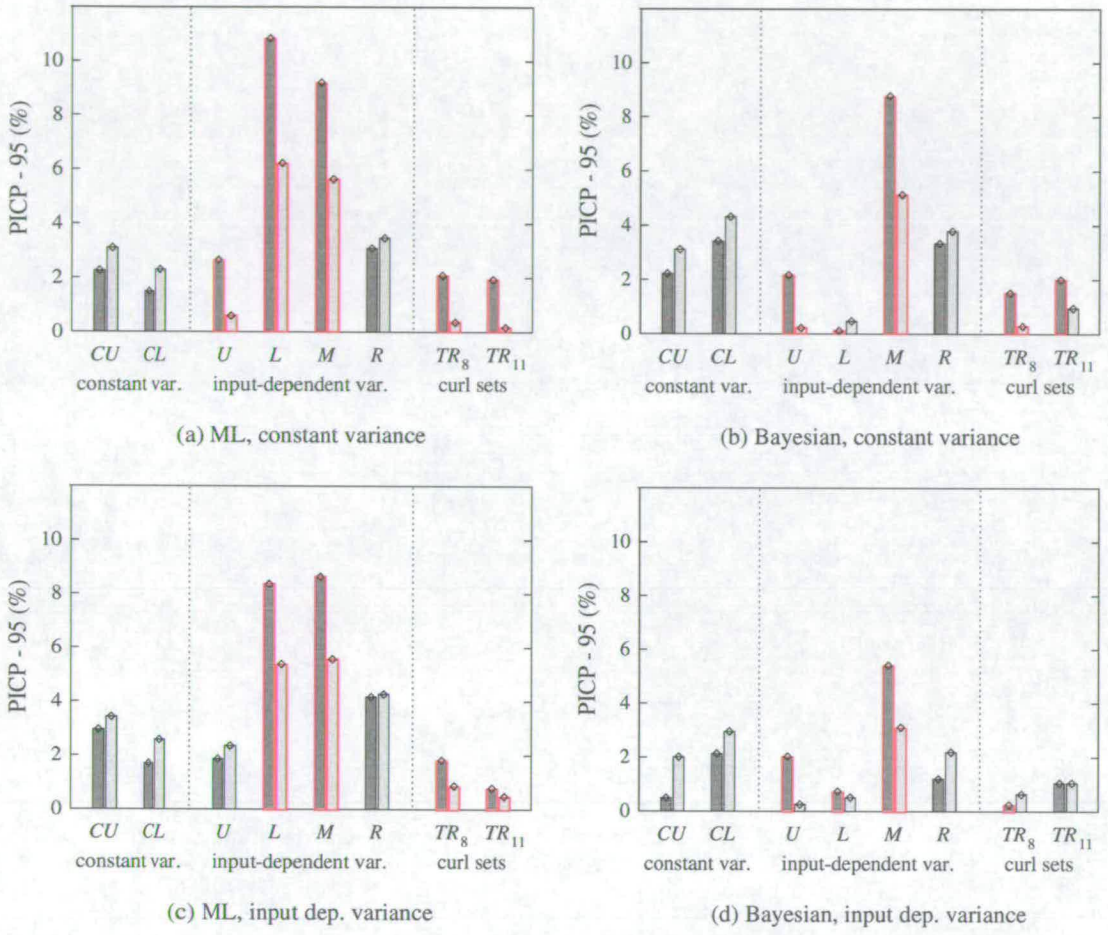


Figure 4.7: *The CI CP results for each CM and artificial data set computed using the exact and approximate delta methods. The absolute deviation of the observed mean CP from the nominal CP (95%) is shown as boxes. Red colour boxes indicate negative difference (observed value smaller than the nominal). The table shows the average absolute deviation for all sets.*



(a) ML, constant variance

(b) Bayesian, constant variance

(c) ML, input dep. variance

(d) Bayesian, input dep. variance

average absolute deviation

		(a)	(b)	(c)	(d)
Exact		4.2	3.0	3.8	1.8
Appr.		2.7	2.3	3.1	1.6

Figure 4.8: *The PI CP results for each CM and artificial data set computed using the exact and approximate delta methods. The absolute deviation of the observed mean CP from the nominal CP (95%) is shown as boxes. Red colour boxes indicate negative difference (observed value smaller than the nominal). The table shows the average absolute deviation for all sets.*

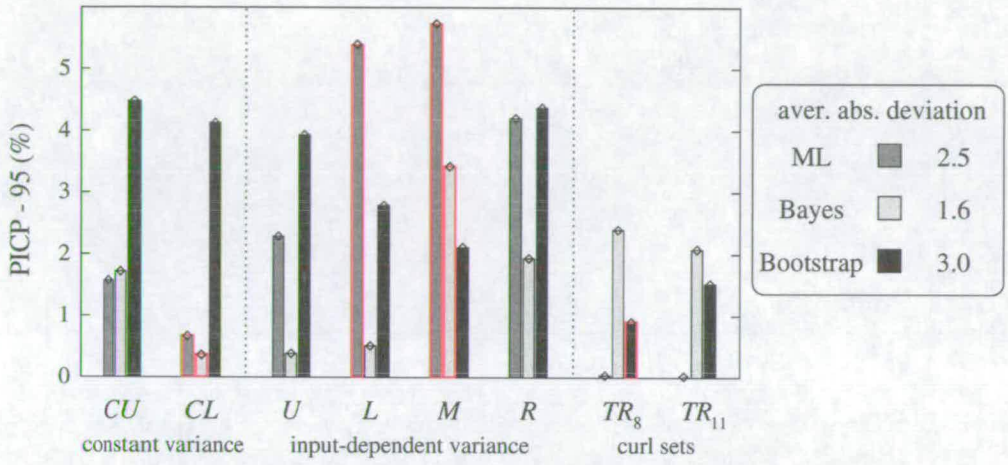
4.5.2 Comparison of the percentage of covered targets to the estimated coverage probability

As mentioned previously, there exist two methods for computing the observed mean PI CP:

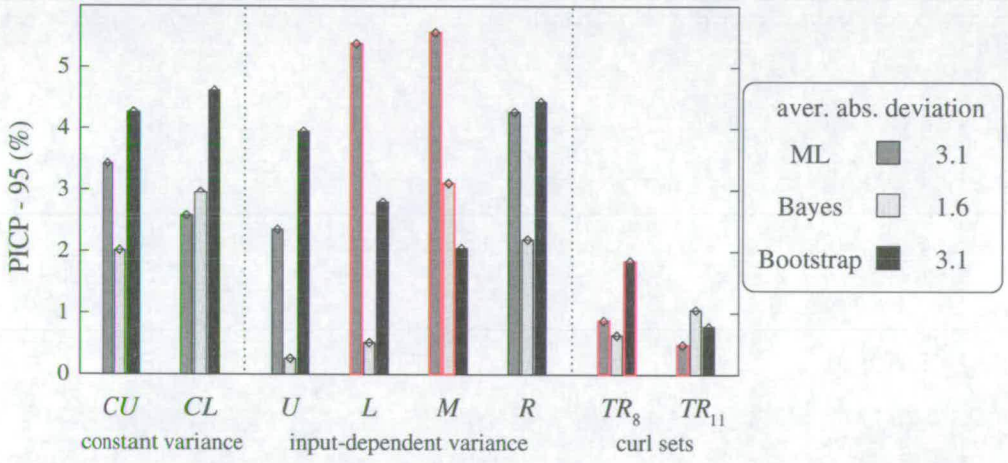
1. The observed mean PI CP can be computed as the percentage of target values that lie within the interval.
2. Alternatively, the estimated CP method can be used, This method is more complex but leads to the estimation of both PI CP mean and standard deviation over the test patterns.

As far as the mean CP is concerned, the two methods can be shown to be almost identical in performance. Figure 4.9 presents the results using the two alternative methods. The difference in average absolute deviation (observed minus nominal CP value) between the two methods is negligible (see tables in fig. 4.9).

Therefore, in the rest of this chapter, the PI CP is computed using the estimated CP method which has the additional advantage of allowing the CP standard deviation to be computed.



(a) Mean PICP computed as the percentage of covered targets



(b) Mean PICP computed using the estimated CP

Figure 4.9: *The observed mean PI CP computed using two alternative methods: (a) the percentage of covered targets and (b) the estimated CP. The absolute deviation of the observed mean CP from the nominal CP (95%) is shown as boxes. Red colour boxes indicate negative difference (observed value smaller than the nominal). The tables show the average absolute deviation for all sets.*

4.5.3 Comparison of committees to individual networks

It is well known that a committee of networks generally has better generalization performance than the individual networks on average. It would be interesting to compare the CM performance of committees to that of individual networks. For the non-simulation methods (ML and Bayesian), it can be expected that the data noise variance estimate averaged over the committee members will be superior to the average estimate of the individual networks. However, as mentioned in chapter 2, using committees introduces an extra uncertainty source, the uncertainty due to the fact that committee members disagree to a certain degree about the output. This uncertainty, called committee uncertainty, is given by

$$\hat{\sigma}^2_{\text{COM}}(\mathbf{x}) = \sum_{i=1}^{M} [y_i(\mathbf{x}) - y_{\text{COM}}(\mathbf{x})]^2/(M-1) \tag{4.17}$$

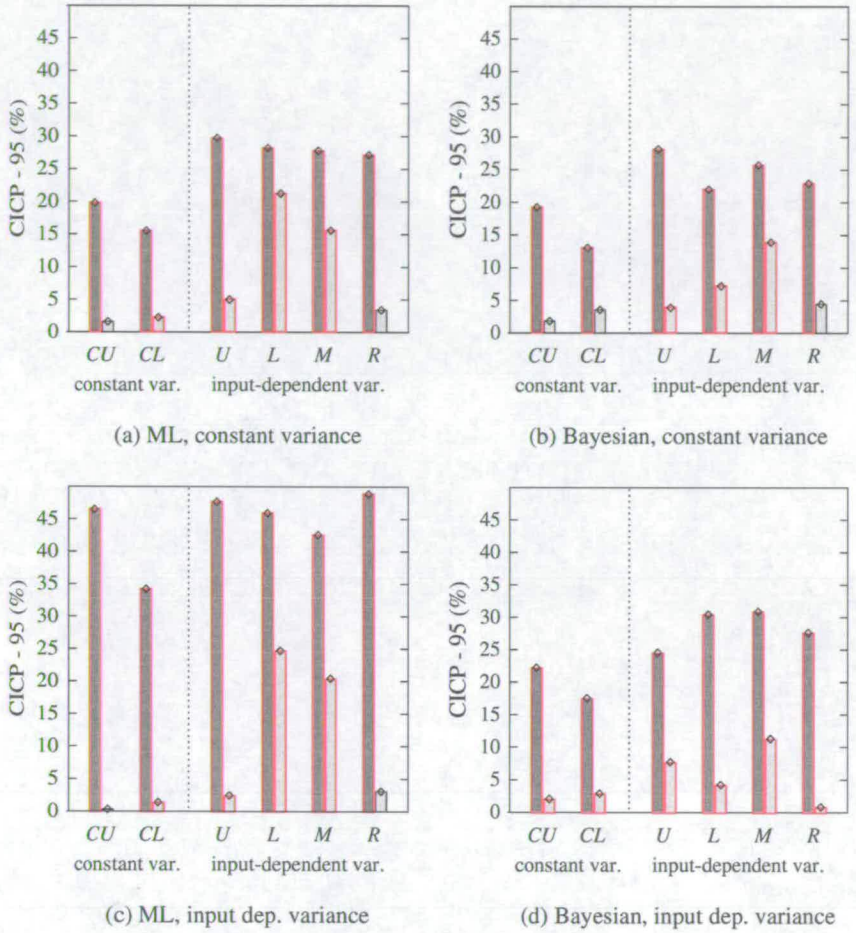
Therefore, this comparative study seeks to establish whether the introduction of an extra uncertainty source has any detrimental effect over the committee CM performance.

Figure 4.10 shows the CI CP result for all available artificial data sets and the ML and Bayesian techniques using the approximate delta method. Confidence measures considering both constant and input-dependent data noise variance have been used. The total model uncertainty variance estimate for the committee is given by

$$\hat{\sigma}^2_m(\mathbf{x}) = <\sigma^2_{m,i}(\mathbf{x})> + \hat{\sigma}^2_{\text{COM}}(\mathbf{x}) \tag{4.18}$$

where $\sigma^2_{m,i}(\mathbf{x})$ is the model uncertainty variance estimate for network i. Clearly, the committee CI CP is superior to the average individual network performance although the uncertainty estimate for committees comprises the average weight uncertainty estimate and the committee uncertainty. Therefore, using committees improves confidence estimation performance as far as model uncertainty estimation is concerned.

Figure 4.11 shows the observed mean PI CP results. As before, the committee estimates perform better than the individual network estimates in most cases. The only counterexample is sets *CU* and *CL* for which the individual networks perform marginally better than the committee for CMs (a), (b) and (d). These results demonstrate that, in most cases, the use of committees improves rather than degrades CM performance for the ML and Bayesian techniques.



(a) ML, constant variance

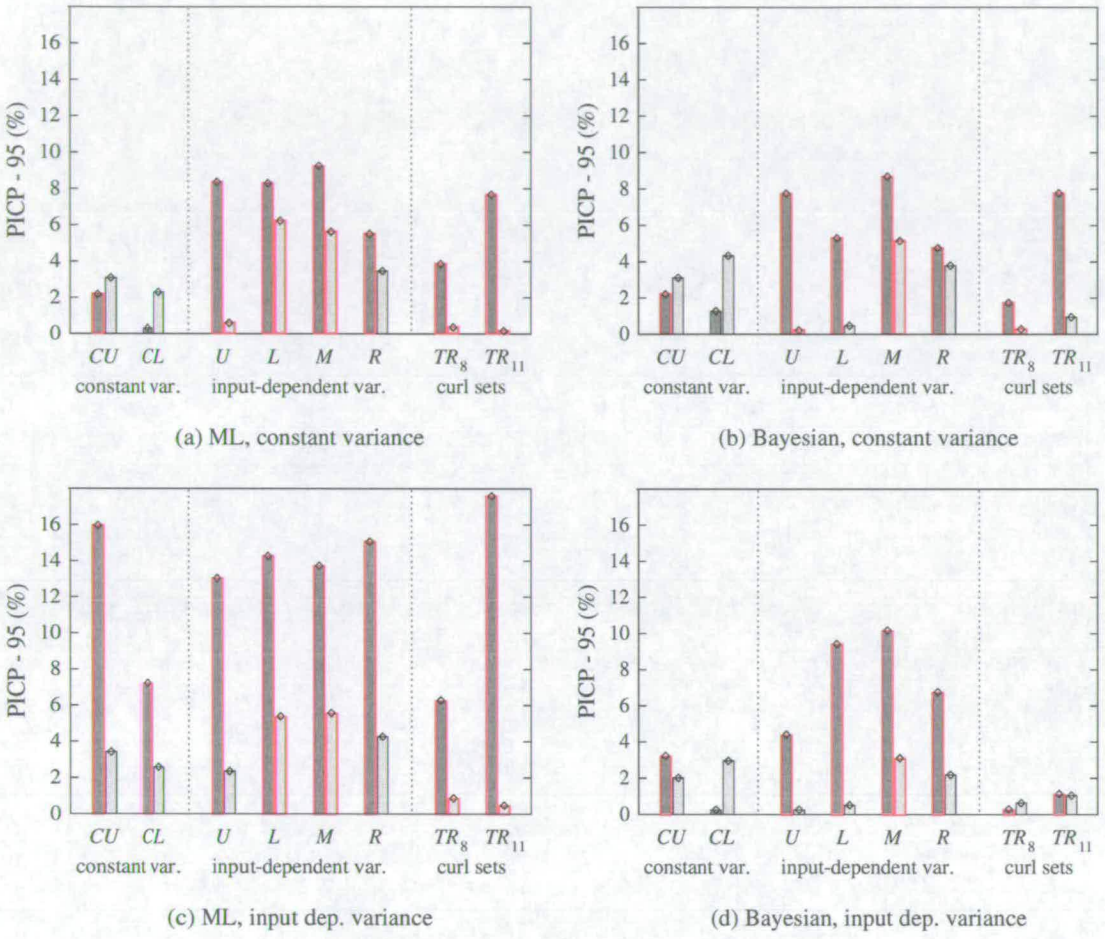
(b) Bayesian, constant variance

(c) ML, input dep. variance

(d) Bayesian, input dep. variance

average absolute deviation	(a)	(b)	(c)	(d)
network	24.7	21.8	44.3	25.5
committee	8.2	5.8	8.7	4.8

Figure 4.10: *The CI CP results for each CM and artificial data set computed as a committee average (white boxes) or an average over individual networks (shaded boxes). The absolute deviation of the observed mean CP from the nominal CP (95%) is shown as boxes. Red colour boxes indicate negative difference (i.e. observed value smaller than the nominal). The table shows the average absolute deviation for all sets.*



(a) ML, constant variance

(b) Bayesian, constant variance

(c) ML, input dep. variance

(d) Bayesian, input dep. variance

average absolute deviation	(a)	(b)	(c)	(d)
network	5.7	4.9	12.9	4.5
committee	2.7	2.3	3.1	1.6

Figure 4.11: *The PI CP results for each CM and data set computed as a committee average (white boxes) or an average over individual networks (shaded boxes). The absolute deviation of the observed mean CP from the nominal CP (95%) is shown as boxes. Red colour boxes indicate negative difference (i.e. observed value smaller than the nominal). The table shows the average absolute deviation for all sets.*

4.5.4 Comparison of CMs that treat data noise variance as constant

This work is an improvement of the comparison study performed in [73]. The CMs are tested on data of both constant and input-dependent actual noise variance. Furthermore, the CI CP and PI CP metrics are used instead of the evaluation approach employed in [73], where only the model uncertainty estimation potential of the methods was tested. The PI CP metric can also be applied to real data as it evaluates the ability to predict targets rather than the true regression. Finally, unlike [73], the results for the non-simulation approaches (ML and Bayesian) were computed using committees of equal size to the bootstrap committee.

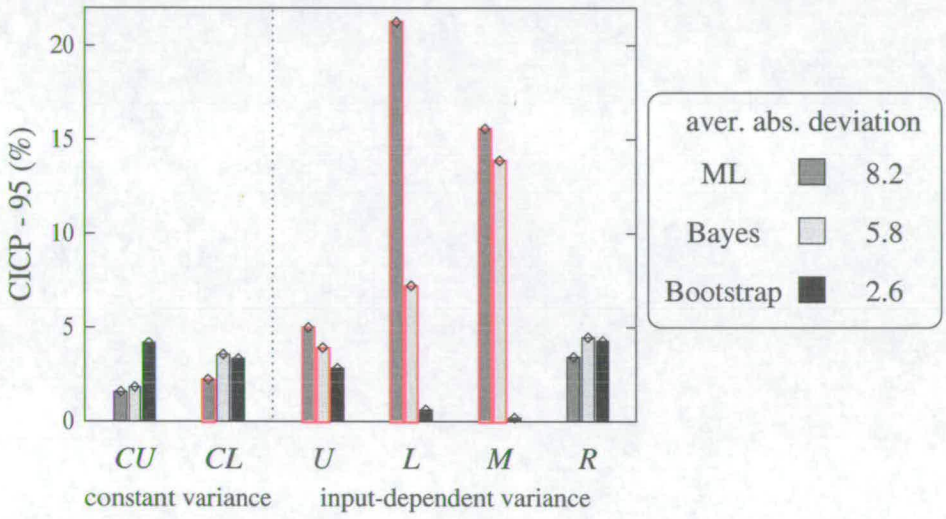


Figure 4.12: *The CI CP results for comparison of CMs that consider constant data noise variance. The absolute deviation of the observed mean CP from the nominal CP (95%) is shown as boxes. Red colour boxes indicate negative difference (i.e. observed value smaller than the nominal). The table shows the average absolute deviation for all sets.*

The results are shown in figures 4.12 and 4.13. Figure 4.12 shows the results for the CI CP estimates. The bootstrap technique yields the most consistent model uncertainty variance estimate. This result agrees with the results of [73]. However, note that the bootstrap result always overestimates the coverage (as indicated by the colour of the boxes in the graph) while the delta result overestimates or underestimates the nominal value. This result is indicative of a possible bias in the bootstrap estimate.

Figure 4.13 shows the PI CP result. The PI CP standard deviation is also shown as an error bar. For clarity, only the upper part of the bar is shown in the graph. The error bars correspond to $\pm 0.5\hat{\sigma}_{\text{cp}}$. Small CP standard deviation indicates that the σ_{TOTAL} estimate is consistent along the

input space. Large CP standard deviation means that $\hat{\sigma}_{\mathrm{TOTAL}}(\mathbf{x})$ often diverges from the actual value locally. The approximate Bayesian technique exhibits the smallest mean CP deviation from the nominal value on average. The bootstrap technique exhibits the best CP standard deviation. Therefore, although the bootstrap technique yields the best model uncertainty variance estimate, it does not exhibit the optimum total standard deviation estimate, as indicated by the PI CP result.

As expected, the Bayesian approach outperforms ML on average, although ML does at least as well for sets *CU*, *M* and *R* for which the PDF is larger in the high noise region. ML also outperforms the Bayesian approach for set *CL* although this set has a smaller PDF in the high noise region. The bias in the ML data noise variance estimate becomes particularly evident for set *L*.

All methods yield observed coverage close to the nominal for the curl sets. Note that the results for the curl sets should be viewed with caution due to the very small test sets.

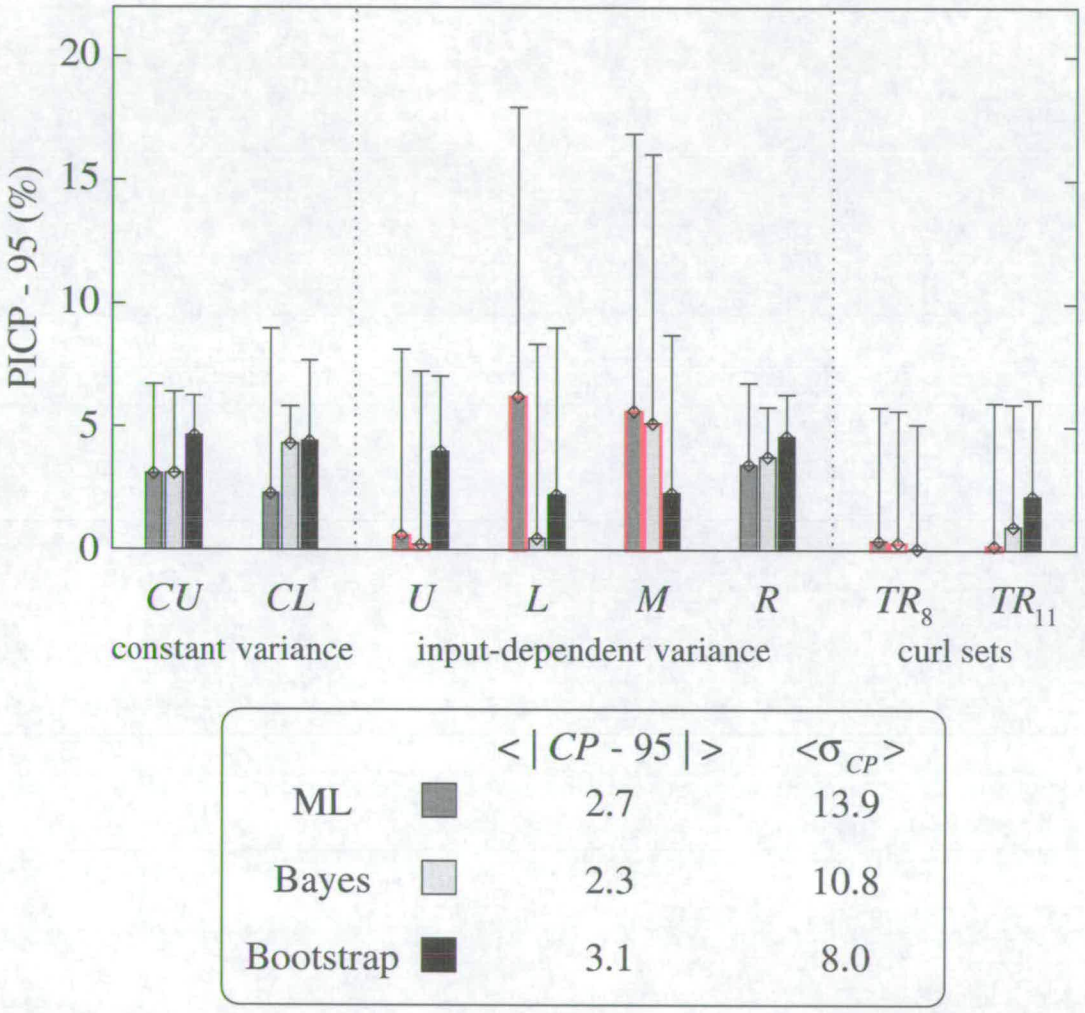


Figure 4.13: *The PI CP results for comparison of CMs that consider constant data noise variance. The absolute deviation of the observed mean CP from the nominal CP (95%) is shown as boxes. Red colour boxes indicate negative difference (i.e. observed value smaller than the nominal). The error bars correspond to* $\pm 0.5\tilde{\sigma}_{CP}$*. The table shows the average absolute deviation and the average CP standard deviation for all sets.*

4.5.5 Comparison of CMs that treat data noise variance as input-dependent

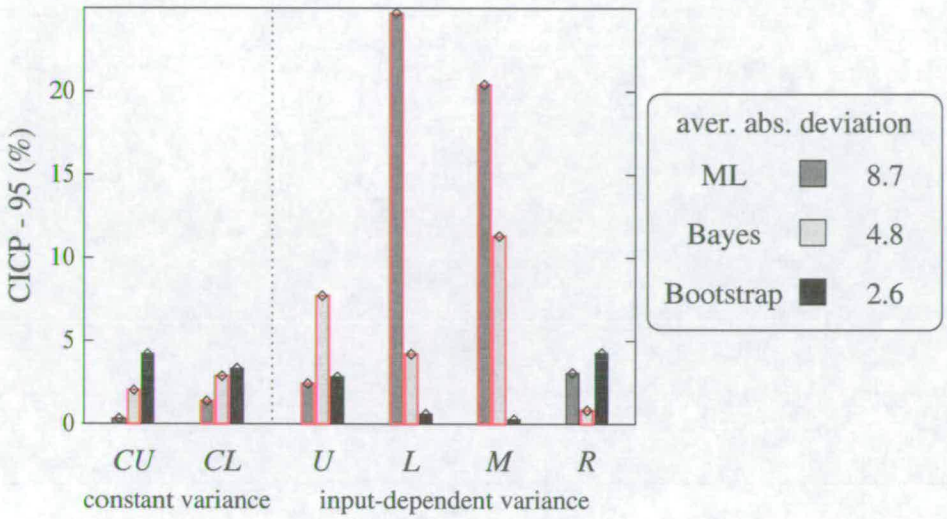


Figure 4.14: *The CI CP results for comparison of CMs that treat data noise variance as input-dependent. The absolute deviation of the observed mean CP from the nominal CP (95%) is shown as boxes. Red colour boxes indicate negative difference (i.e. observed value smaller than the nominal). The table shows the average absolute deviation for all sets.*

In this comparison study all the CMs assume that data noise variance is input-dependent. Figure 4.14 summarises the CI CP results. As in the case of CMs that treat data noise variance as constant (section 4.5.4 and [73]), the bootstrap method outperforms the delta method on average in terms of model uncertainty estimation. Again, the bootstrap method always overestimates the coverage while the delta method overestimates or underestimates the nominal value.

Figure 4.15 shows the PI CP results. The most consistent mean CP estimate is obtained using the approximate Bayesian approach for regression and σ_ν^2 estimation. Again, the bootstrap technique exhibits the smallest CP standard deviation. However, on average, it overestimates the mean coverage for all sets. The two augmented network approaches yield similar CP standard deviation. As previously, the Bayesian approach outperforms ML on average, especially for set L for which the PDF is low in the high noise variance region and the bias in the ML prediction becomes more apparent. Again, all methods achieve observed coverage close to the nominal for the curl sets.

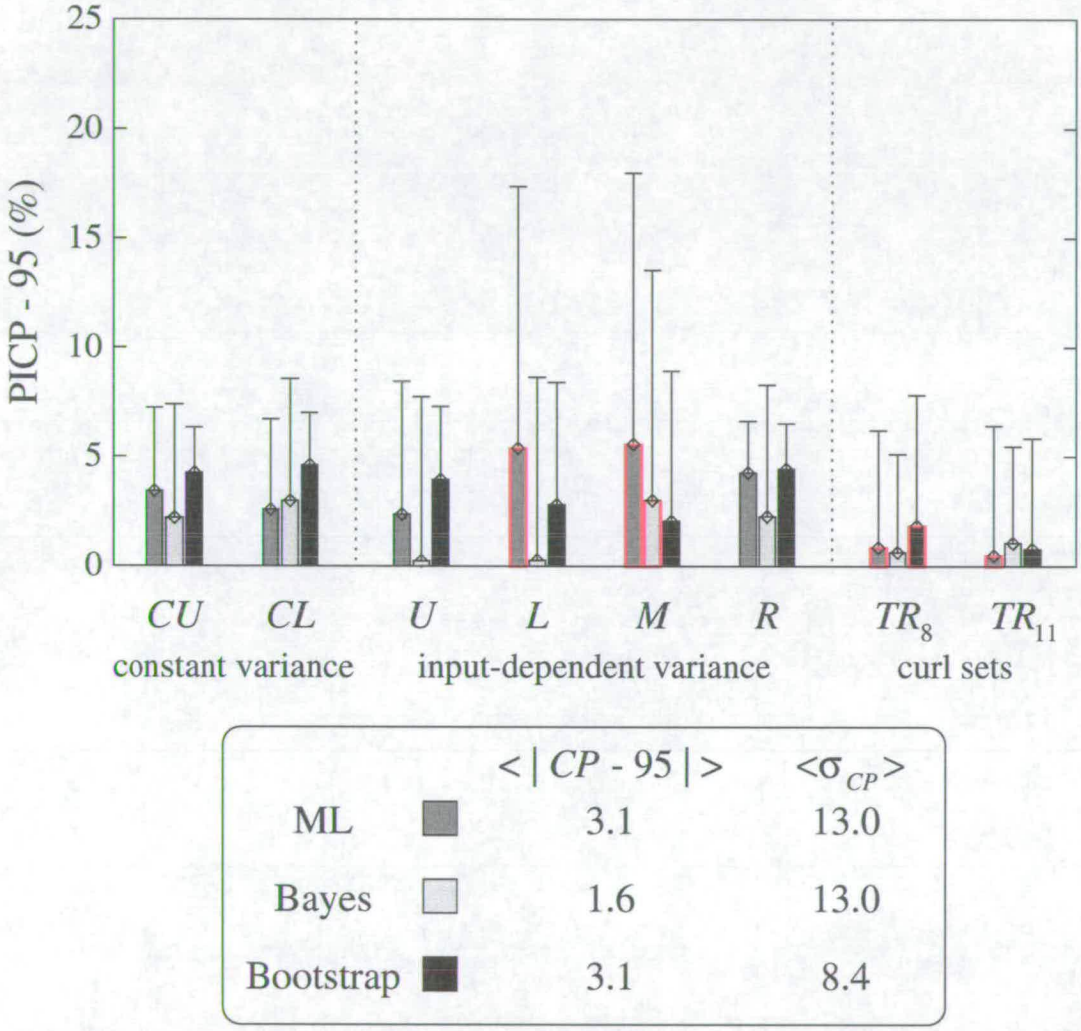


Figure 4.15: *The PI CP results for comparison of CMs that treat data noise variance as input-dependent. The absolute deviation of the observed mean CP from the nominal CP (95%) is shown as boxes. Red colour boxes indicate negative difference (i.e. observed value smaller than the nominal). The error bars correspond to* $\pm 0.5\hat{\sigma}_{cp}$*. The table shows the average absolute deviation and the average CP standard deviation for all sets.*

4.5.6 Comparison of constant to input-dependent noise variance CMs

In [22] the noise in the curl data was treated as constant over input space. Here, the effect of treating noise variance as a function of the inputs is investigated. The actual nature of the curl data noise variance is, naturally, unknown. In linear least-squares regression there exist various tests for detecting heteroscedasticity, *i.e.* noise variance that is non-constant over the input space [66]. In a neural network context, Qazaz [17] shows that for a particular real problem improved generalization performance is achieved when the augmented network model is used. This suggests that the noise in the data has input-dependent variance.

As far as the curl sets are concerned, the test MSE results are contradictory. As shown in table 4.1, the ML augmented network committees achieve better test MSE than the corresponding constant variance committees for test TR_8 and the Bayesian augmented committees achieve better MSE than the baseline for test TR_{11}. In the other two cases, (Bayesian committees for

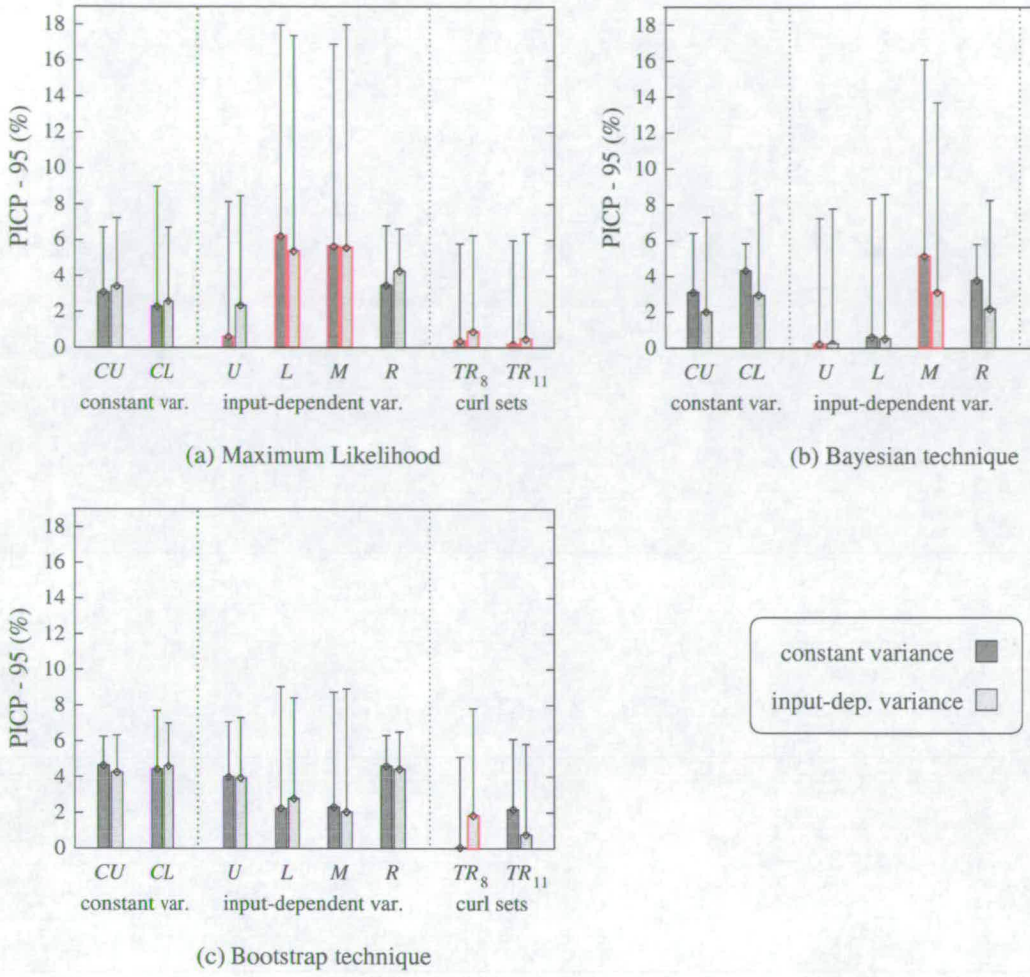
Data Set	ML baseline	ML augmented	Bayesian baseline	Bayesian augmented
TR_8	122.4	121.7	121.8	123.6
TR_{11}	168.3	169.3	169.1	166.6

Table 4.1: *Average test MSE for the curl sets using the baseline (constant data noise variance) and the augmented network methods.*

set TR_8 and ML committees for set TR_{11}) the baseline approach outperforms the augmented approach. Therefore, no safe conclusion can be reached regarding the actual form of the data noise in the curl data. In any case, it would be dangerous to base any conclusion on generalization performance alone since the test sets for the two curl sets contain only 185 and 224 examples and determining the best possible generalization performance for each method would require a more principled training approach (*e.g.* trying different regularization parameter values for each method) than the one followed here.

Therefore, it would be interesting to investigate whether any observable change in confidence estimation performance occurs when the input-dependent variance approach is used. If this is the case then it may be possible to infer the input-dependency of the data noise variance of the curl sets by looking at the coverage result of the two approaches.

For this reason, the performance (in terms of observed PI CP) of confidence estimation methods that assume constant noise variance is compared to that of CMs that treat noise variance as



(a) Maximum Likelihood

(b) Bayesian technique

(c) Bootstrap technique

Figure 4.16: *The absolute deviation of the observed mean CP from the nominal value and the CP standard deviation for the three CMs, treating σ_ν^2 as a constant (shaded boxes) or a function of the inputs (unshaded boxes). Red colour boxes indicate negative deviation (i.e. observed value smaller than the nominal). The error bars correspond to $\pm 0.5\hat{\sigma}_{cp}$.*

input-dependent. Artificial data sets of both constant and input-dependent true noise variance are used to investigate whether a CM exhibits better CP when applied to data whose noise form matches the adopted assumption. In other words, if the PI CP mean and standard deviation were sensitive to the true input-dependency of the data noise variance, then methods that assume that the data noise variance is constant would outperform methods that assume input-dependent noise for sets which were created using noise of constant variance. Likewise, CMs that treat noise variance as input-dependent would outperform constant-variance CMs for sets which were created using input-dependent noise.

Fig. 4.16 shows the absolute mean CP deviation (observed minus nominal value) and the CP

standard deviation for all the data sets using
(a) the ML σ_ν^2 estimate and the approximate delta method,
(b) the approximate Bayesian σ_ν^2 estimate and the approximate delta method, and
(c) the bootstrap technique.
The dark shaded boxes correspond to CP estimates obtained under the constant σ_ν^2 assumption and the light shaded boxes to estimates obtained under the input-dependent σ_ν^2 assumption. Evidently, it is not possible to infer the actual input-dependency of the data noise variance by looking at the observed mean CP performance. For instance, using CM (b) the constant σ_ν^2 methods achieve worse mean CP for sets *CU* and *CL*, the sets created using constant σ_ν^2. Moreover, occasionally the constant σ_ν^2 methods outperform the input-dependent σ_ν^2 methods for sets which were created using input-dependent σ_ν^2. For example, set *U* using CM (a) and set *L* using CM (c). Moreover, in many other cases (for example, CM (a) and set *M*, CM (b) and sets *U*, *L*, and CM (c) and sets *CL*, *U*, *R*) the observed mean CP obtained by either approach (treating σ_ν^2 as constant or input-dependent) is almost identical.

The failure of the mean CP to discriminate between data sets of constant and input-dependent noise level was more or less anticipated. The reason is that, as an average over all test patterns, the observed mean PI CP may be close to the nominal value even if the σ_{TOTAL} estimate is locally poor. This is because locally bad effects may compensate for one another, as explained in section 4.3.4.

The standard deviation of the CP over the test patterns $\hat{\sigma}_{\text{cp}}$ is a measure of local performance. Large $\hat{\sigma}_{\text{cp}}$ indicates that the uncertainty estimate is locally bad. For example, the result for set *L* using the bootstrap technique (fig. 4.16 (c)) can be explained in terms of local confidence estimation performance. The constant σ_ν^2 approach outperforms the input-dependent σ_ν^2 approach in terms of mean CP, however, it exhibits larger CP standard deviation. This implies that the constant variance $\hat{\sigma}_{\text{TOTAL}}$ overestimates the true standard deviation for some test patterns while underestimating it for some others. These locally bad estimates compensate for one another, giving a misleading, good mean CP estimate. However, the constant σ_ν^2 approach yields larger $\hat{\sigma}_{\text{cp}}$ even for constant variance sets, especially set *CL* and CMs (a) and (c). Consequently, the CP standard deviation can not reliably gauge local CM performance.

actual variance	const. var. CM ■	input-dep. var. CM □
constant	2.7 (10.3)	3.0 (8.0)
input-dep	4.0 (16.9)	4.4 (16.5)
curl	0.3 (11.2)	0.7 (11.3)

(a) Maximum Likelihood

actual variance	const. var. CM ■	input-dep. var. CM □
constant	3.7 (4.8)	2.5 (10.9)
input-dep.	2.4 (13.9)	1.5 (16.1)
curl	0.6 (10.3)	0.8 (8.9)

(b) Bayesian technique

actual variance	const. var. CM ■	input-dep. var. CM □
constant	4.5 (4.9)	4.4 (4.5)
input-dep.	3.3 (9.0)	3.3 (9.0)
curl	1.1 (9.0)	1.3 (11.0)

(c) Bootstrap technique

Table 4.2: *The average absolute deviation of the observed mean PI CP from the nominal value and average PI CP standard deviation for the comparison of CMs that treat noise variance as constant to CMs that consider input-dependent noise variance.*

This result is summarised in table 4.2 in which the average values of observed CP deviation from the nominal value and CP standard deviation are shown separately for sets of constant (*CU* and *CL*), input-dependent (*U*, *L*, *M*, and *R*) and unknown (TR_8 and TR_{11}) true data noise variance. The performance of CMs that treat noise variance as constant or input-dependent does not depend on the actual input-dependency of the noise variance. Therefore, neither the observed mean CP nor the CP standard deviation can be used to reliably distinguish constant from input-dependent σ_ν^2 sets.

Conclusions

As explained in section 4.3.4 adopting the wrong assumption about the input-dependency of data noise variance leads to locally inaccurate confidence estimates. It was shown that the PI CP can only assess the overall performance of a CM (*i.e.* averaged over the entire input space) and should not be used to investigate region-dependent effects such as the input-dependency of the noise in the data. In other words, the PI CP cannot be used to investigate the local quality of the confidence estimates.

Following Qazaz [17], the test MSE could be used as a criterion for determining whether the input-dependency of the curl data noise. However, the test MSE results achieved by augmented and baseline methods for the curl sets are contradictory and no safe conclusion can be reached.

4.6 Chapter summary and conclusions

This chapter presented previous CM comparison studies as well as methods for evaluating CM performance. CM performance was investigated on artificial and real data sets using the common method of computing the observed CI or PI coverage probability. Ideally, the observed mean CP should be near to the nominal value (95% in the case of these experiments). Using only the deviation of the observed mean CP from the nominal is inadequate since the mean CP can only gauge estimation capability as an average over the input space. The local quality of the σ_{TOTAL} estimate can be investigated by computing the CP standard deviation over the test points. The CP standard deviation can be computed by first estimating the CP at each test point.

The delta method with the approximate Hessian was found to be a more consistent estimator than the delta method that uses the exact Hessian. This result agrees with results of previous studies [15]. Computation and inversion of the exact Hessian is sometimes unstable. This

instability inevitably affects the performance of the exact delta estimator. As far as the non-simulation methods are concerned, the confidence estimation performance of a committee of networks was found to be superior to that of the average individual network in the vast majority of the cases.

The best observed mean CP estimate is given by the Bayesian approach with the approximate delta method. As expected, the approximate Bayesian approach generally yields better coverage than ML, because, unlike ML, it yields an unbiased noise variance estimate. The second best observed mean CP estimate is given by the bootstrap method which also exhibits the smallest CP standard deviation and a consistent overestimation of prediction variance. There is no significant difference between the augmented network approaches in terms of CP standard deviation.

Overall conclusion

The *local* quality of CM performance is particularly important in real ANN applications. Pragmatically, a CM must be sensitive to the density of the input data and the data noise. Specifically, it is crucial that high noise regions and/or regions of low training data PDF are distinct from low noise regions and/or regions of high data PDF. This will be true only if the CM yields good local estimates of uncertainty. It is well known that adopting an erroneous assumption about the form of noise variance significantly deteriorates local confidence estimation performance. However, it was found that the performance - in terms of both CP mean and CP standard deviation - of constant and input-dependent data noise variance CMs does not match the actual input-dependency of the noise in the data. Therefore, it was concluded that metrics based on the PI CP cannot be used to investigate the local quality of CM performance. Different approaches to CM evaluation must be sought to investigate local CM performance.

Chapter 5
A region-dependent approach to assessing confidence estimation performance

5.1 Introduction

The results of chapter 4 showed that the PI CP metric is not appropriate for evaluating quality of the local performance of confidence estimates. In this chapter, a new, region-dependent approach to confidence measure evaluation is proposed. The new approach, called *classification of local uncertainty estimates*, manages to capture the local aspect of CM performance.

The chapter is structured as follows.

Section 5.2 is an introduction to the concept of region-dependent CM evaluation.

Section 5.3 describes the CM evaluation algorithm in detail.

Section 5.4 presents the experimental setup and the heuristic algorithms used to partition the input space and define the confidence regions for artificial and real sets.

Section 5.5 presents results of two comparison studies using the new approach. First, CMs that treat noise variance as constant are compared to CMs that treat noise variance as input-dependent. The second study compares CMs that use the augmented network architecture (ML and Bayesian techniques) to CMs that use the conventional network architecture (the bootstrap technique).

5.2 Region-dependent CM evaluation

The previous chapter showed that the commonly used PI CP metric for CM performance evaluation is inadequate for investigating local performance. The reason is that the PI CP treats the entire input space as a whole. Therefore, local effects are lost in the averaging. In contrast,

classification of local uncertainty estimates (CLUES) explicitly quantifies the ability of a CM to distinguish regions of interest in the input space, *i.e.* local performance.

The CLUES technique aims to meet the practical requirements of a real-world confidence estimation system. Such a system should be able to *distinguish* between *reliable* and *unreliable* predictions. This can only be achieved if the CM is able to associate high variance with inputs originating from *high noise, low data density* regions and low variance with inputs originating from *low noise, high data density* regions. A CM must therefore yield good local estimates of uncertainty (*i.e.* locally good estimates of the prediction standard deviation). The CLUES technique can be used to assess (and compare) CMs on the basis of their *local* performance, rather than their *average* performance over input space.

The rest of this section highlights the basic concepts on which the CLUES algorithm is based.

5.2.1 Partitioning the input space

A confidence measure can be viewed as a relative measure of the usefulness of the prediction. Prediction confidence for a given test pattern is determined largely by the nature of the neighbourhood of the input space in which it belongs, as a confidence measure comprises an input-dependent (in the general case) data noise variance estimate and a model uncertainty measure (which depends on the training data density) (*e.g.* [19, 23, 30]). The CLUES method is based on the idea of partitioning the input space and treating each partition as an independent region of interest.

If an independent, previously unseen, test set is available and the architecture is assumed to be optimised, the average *absolute prediction error*[1] $<r(\mathbf{x})>= <|y(\mathbf{x}) - t|>$ for test points that belong to a particular region of the input space is indicative of the local error and the local uncertainty level. In theory the absolute prediction error contains contributions of both sources of uncertainty. However, typically, in regions represented in the training data, the PDF is relatively high and the main contribution to the error is data noise variance. For example, this assumption is implicitly adopted when data noise variance is estimated using the squared prediction errors as targets of a neural network as in [26]. In general, the entire input space can be divided into C partitions (which may be disjoint) and the average $r(\mathbf{x})$ of each partition evaluated. In fig. 5.1, this procedure is demonstrated using a one-dimensional data set. The

[1] In principle, the average MSE could be used instead of the absolute prediction error.

input space is divided into three partitions using two thresholding bounds.

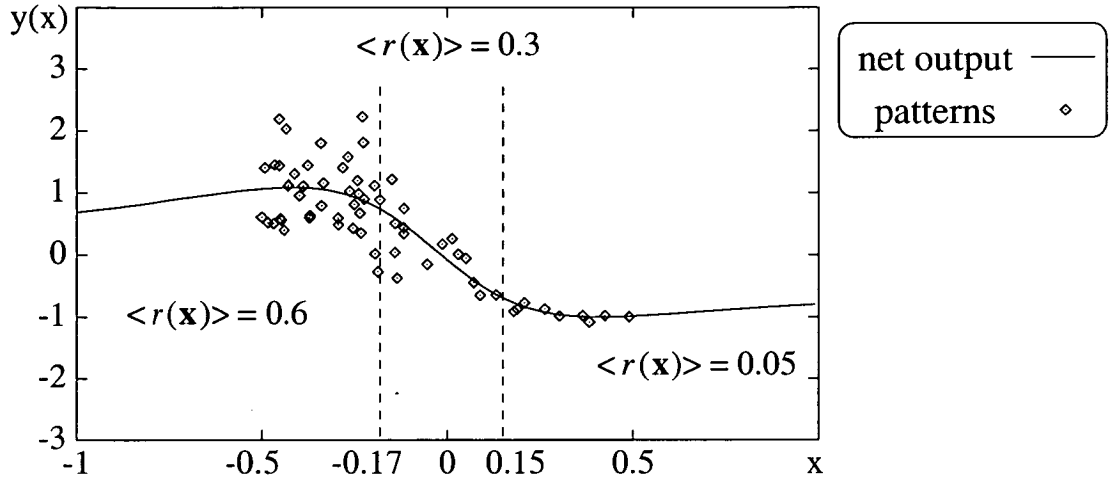


Figure 5.1: *The graph shows an one-dimensional data set and the function obtained by an MLP trained on the data. Two thresholding bounds (dashed lines) are used to divide the input space into three regions with respect to the average absolute prediction error within each region. The average absolute prediction error is given for each partition.*

As shown in fig. 5.1, partitions must take into account the actual distribution of data errors (or equivalently the distribution of prediction error) across the input space, so that each region comprises patterns of similar error magnitude. In a real application, the true distribution of error is not known. However, the nature of the application may provide some additional information about σ_ν^2 and, more importantly, how data errors are distributed in input space. For example, the curl measurement method [22] is more likely to yield large errors when curl is large rather than when it is small. Partitions can therefore be defined accordingly. Input space partitions must be large enough and they must have approximately equal data density so that the average $r(\mathbf{x})$ of the points in each partition is representative of the local uncertainty due to data noise. If partitions are too large, detailed information is averaged and lost. If partitions are too small, the average $r(\mathbf{x})$ will suffer high variance error due to the small data sample. Naturally, only the region of the input space that is represented in the data set is partitioned. New inputs originating from unrepresented regions are detected (and rejected) by a *novelty detection* scheme [18] as explained in chapter 3.

5.2.2 Confidence levels and confidence classes

If all the aforementioned requirements are met, the average $r(\mathbf{x})$ of each partition is characteristic of the uncertainty level for that region of the input space. Therefore, each of the C partitions corresponds to one of a set of C *confidence levels*. Test patterns that belong to a particular partition constitute a *confidence class*, *i.e.* they share the same level of confidence. Thus, the test set is effectively divided into C confidence classes. This scheme naturally leads to a "quantised" confidence measure. The (real valued) prediction standard deviation is quantised into C confidence levels. Thus, each network prediction can be assigned a confidence level according to the magnitude of its associated estimate of standard deviation $\hat{\sigma}_{\text{TOTAL}}$. In other words, the standard deviation estimate is thresholded. For example, a single threshold creates two confidence classes (*e.g.* reliable - unreliable predictions) and two thresholds lead to three confidence classes (*e.g.* high, average and low confidence). The actual threshold values can be computed using the average $r(\mathbf{x})$ of the partitions.

From a practical point of view, a quantised CM (*i.e.* a confidence index) may be preferable to an error bar or standard deviation estimate in many real applications. Representing confidence as an index is more intelligible for the non-experts that are likely to be system operators. During development of the curl estimation system it was found that the machine operators may prefer a confidence index or characterisation (*i.e.* high, low, average confidence) to a real valued confidence measure for reasons of clarity. Therefore, comparing CMs *by quantising* the real valued confidence estimate is consistent with the pragmatic demands of real usage.

5.2.3 Confidence measures as classification mechanisms

Local CM performance can be evaluated by checking how well the standard deviation estimate matches the variation in prediction error in each of the confidence classes. This is achieved by *thresholding* the standard deviation estimate and assigning the prediction to the first confidence class whose characteristic *threshold* is larger than the σ_{TOTAL} estimate. In other words, the CM is treated as a *classification* mechanism and local performance is quantified by computing the classification error.

Naturally, partitioning the input space and averaging over each partition causes some detailed information to be lost. On the other hand, this averaging is a simple way to gauge the local performance of a CM. This concession is necessary as the notion "locality" is implicit in MLPs.

In contrast, in *Radial Basis Function* networks locality is well defined and easily incorporated into confidence estimation [132, 133].

In the following section the CLUES algorithm is described in detail.

5.3 The CLUES Algorithm

The CLUES technique uses a CM to classify predictions into one of C confidence classes. Each confidence class corresponds to one of C partitions of the input space called *confidence regions*. If a confidence region contains patterns of similar noise level and the average PDF is approximately equal for all regions, predictions for patterns belonging to the same region should be associated with similar standard deviation estimates. For an artificial data set, partitions of the input space can be created within which the PDF and noise variance of the training data are approximately equal for all patterns $(\mathbf{x}, t)$ that belong to region i. For example, in this chapter, the hyper-planes that separate the regions for the artificial data have been chosen by thresholding the principal direction of variation for σ_ν^2 and PDF. The details of the partitioning approach for artificial data and the real, curl data will be discussed in section 5.4. For the moment, it is assumed that the confidence regions and classes have been defined.

If the C partitions are defined with respect to the distribution of data errors and the average PDF is approximately equal for each partition, the average absolute prediction error of each region exhibits a trend which characterises the actual error distribution. Therefore, each region corresponds to a certain confidence level and the confidence index associated with a new input is determined by the region to which it belongs. The average $r(\mathbf{x})$ of the points in partition i is denoted by r_i. The region indices are sorted in order of increasing r_i so that r_1 corresponds to the region of minimum average prediction error (*i.e.* highest confidence) and r_C to the maximum prediction error (*i.e.* minimum confidence) region. For a given test set, a *confidence class* C_i is defined as the set that contains all test patterns that lie in region i:

$$C_i = \{\mathbf{x}^k : \mathrm{c}(\mathbf{x}^k) = i\} \qquad k = 1, ..., N_t \tag{5.1}$$

where $\mathrm{c}(\mathbf{x})$ is a function that returns the index i of the confidence region in which pattern $\mathbf{x}$ belongs and N_t is the number of test patterns.

5.3.1 Choosing thresholds for the confidence classes

In order to classify the confidence estimates, each confidence class C_i must be assigned a characteristic standard deviation threshold s_i. Since classes are sorted in order of increasing r_i, if $i<k$ then $r_i<r_k$ and it follows that $s_i<s_k$.

In reality, the choice of thresholds will seriously affect the classification outcome. However, the aim is solely to achieve a *relative* CM comparison on the basis of local performance. In other words, it is the relative classification error for different CMs rather than its *absolute* value that matters. Therefore, simple heuristic rules can be used to choose class thresholds.

The heuristic choice of thresholds used for the results of this chapter is based on the observation that the average standard deviation estimate should be approximately proportional to r_i for each region. Therefore, threshold s_i separating region i from region $i+1$ must be proportional to the average of r_i and r_{i+1}:

$$s_i = \frac{<\hat{\sigma}(\mathbf{x})>}{<r(\mathbf{x})>} \cdot \frac{r_i + r_{i+1}}{2} \qquad \text{if} \qquad 0 < i < C \tag{5.2}$$

where $<\hat{\sigma}(\mathbf{x})>$ is the average over all test points of the total standard deviation estimate $\hat{\sigma}_{\text{TOTAL}}(\mathbf{x})$ and $<r(\mathbf{x})>$ the average over all test points absolute prediction error. Thresholds s_0 and s_C are defined to have values 0 and ∞ respectively. A schematic illustration of eq. (5.2) is shown in fig. 5.2.

The term $<\hat{\sigma}(\mathbf{x})>/<r(\mathbf{x})>$ serves as a scaling factor. It ensures that the class thresholds are of the same magnitude as the σ_{TOTAL} estimates, since the contributions from both uncertainty sources to $\hat{\sigma}_{\text{TOTAL}}(\mathbf{x})$ render the average magnitude of $\hat{\sigma}_{\text{TOTAL}}(\mathbf{x})$ typically larger than that of $r(\mathbf{x})$. The average standard deviation $<\hat{\sigma}(\mathbf{x})>$ is computed separately for each CM. In practice, two (or more) regression models may be available, *i.e.* the regression models of the CMs under comparison. As the true regression function is unknown, it is reasonable to compute the average over the regression models $<r(\mathbf{x})>$ and r_i's, and use these averaged values in eq. (5.2) to compute the thresholds.

In this chapter, local uncertainty is evaluated by thresholding the total standard deviation estimate $\hat{\sigma}_{\text{TOTAL}}(\mathbf{x}) = \sqrt{\hat{\sigma}_{\nu}^2(\mathbf{x}) + \hat{\sigma}_{m}^2(\mathbf{x})}$. An alternative would be to threshold directly the data noise standard deviation estimate $\hat{\sigma}_{\nu}(\mathbf{x})$. This would alleviate the need for a scaling factor in eq. (5.2). However, thresholding the total standard deviation is preferred in order to evaluate

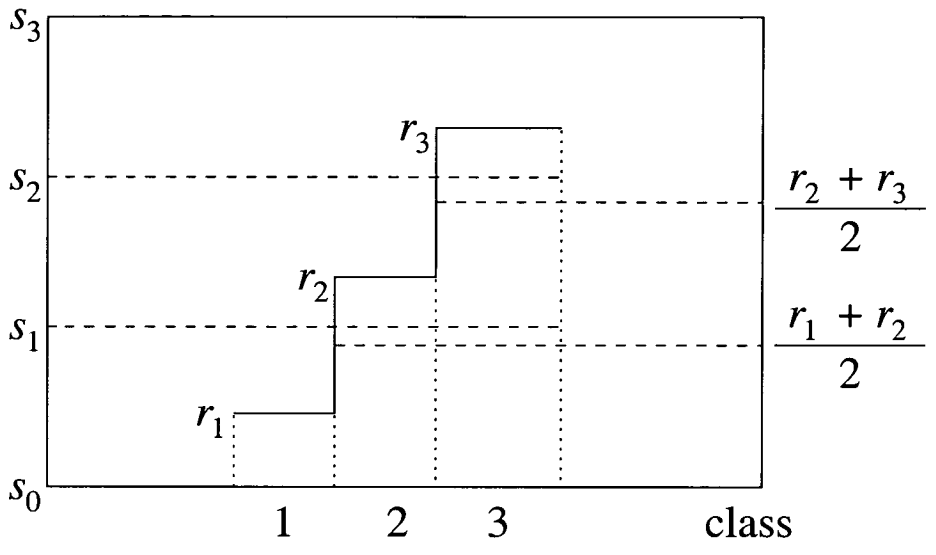


Figure 5.2: *Schematic illustration of the heuristic algorithm for the choice of thresholds for the confidence classes that correspond to the confidence regions shown in fig. 5.1. Thresholds s_0 and s_3 are defined to have values of 0 and ∞ respectively.*

and compare the performance of CMs as complete confidence estimation components.

5.3.2 Classification of confidence estimates

The next step is to use the thresholds to classify confidence estimates. The real valued confidence estimate $\hat{\sigma}_{\text{TOTAL}}(\mathbf{x}^k)$ is compared with the class thresholds and pattern $\mathbf{x}^k$ is classified to confidence class i according to the following rule:

$$\mathbf{x}^k \in \mathcal{S}_i \qquad \textit{iff} \qquad s_{i-1} < \sigma_{\text{TOTAL}}(\mathbf{x}^k) < s_i \tag{5.3}$$

where $\mathcal{S}_i$ is the set that contains all points classified as belonging to class $\mathcal{C}_i$.

Thus, set $\mathcal{S}_i$ contains all points whose $\hat{\sigma}_{\text{TOTAL}}$ is smaller than s_i but larger than s_{i-1}. Considering the prediction probability distribution $p(t|y(x))$ (see fig. 5.3) it is evident that, effectively, pattern $\mathbf{x}^k$ is classified in class i if and only if the probability that the true value lies in the interval

$$(y(\mathbf{x}^k) - s_i, y(\mathbf{x}^k) + s_i)$$

is greater than 68% and the probability that the true value lies in

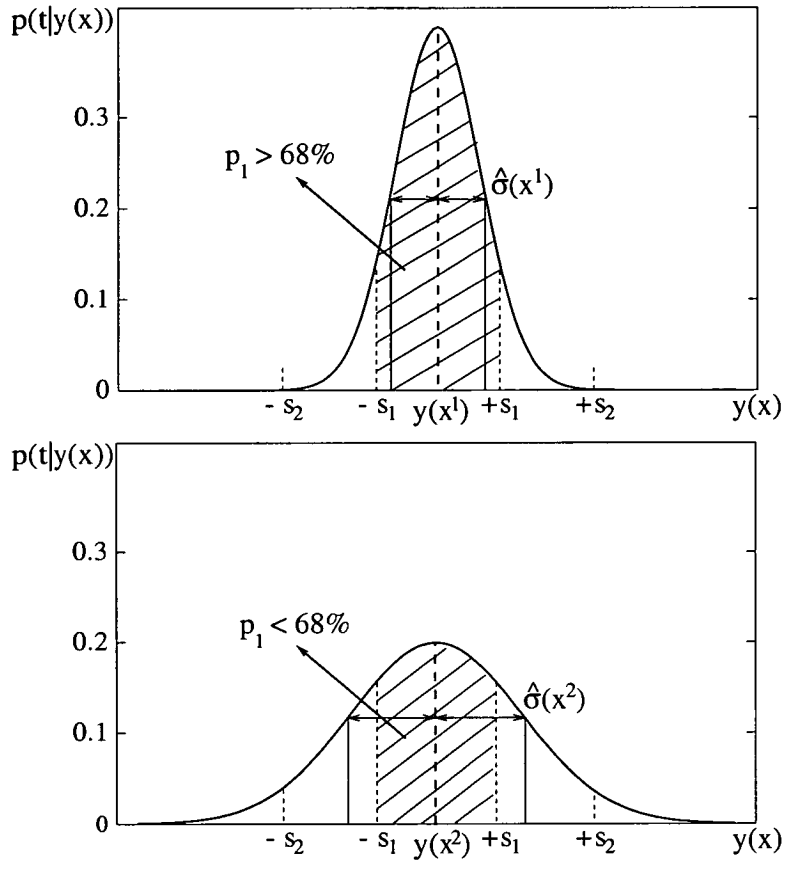


Figure 5.3: *Schematic illustration of the thresholding principle used to classify predictions according to the size of their standard deviation. Pattern x^1 is assigned to class 1, pattern x^2 to class 2.*

$$(y(\mathbf{x}^k) - s_{i-1}, y(\mathbf{x}^k) + s_{i-1})$$

is smaller than 68%. This is because the true value is known to lie in the interval

$$(y(\mathbf{x}^k) - \hat{\sigma}(\mathbf{x}^k), y(\mathbf{x}^k) + \hat{\sigma}(\mathbf{x}^k))$$

68% of the time. Thus,

$$\hat{\sigma}(\mathbf{x}^k) < s_i$$

implies that the probability p_i of the true value lying in interval $(y(\mathbf{x}^k) - s_i, y(\mathbf{x}^k) + s_i)$ is greater than 68%. Similarly,

$$\hat{\sigma}(\mathbf{x}^k) > s_i,$$

implies p_i<68%.

A confidence measure may not classify all or even most of the test points correctly. The standard deviation of $r(\mathbf{x})$ around its mean r_i for patterns in C_i is substantial and patterns near the boundaries may be misclassified. Moreover, it is difficult to choose the optimum thresholds for the classes. For these reasons, the classification error is typically large. However, as already mentioned, the magnitude of the classification error does not present a problem since the aim is to obtain a *relative* measure of local CM performance.

For each test pattern, the correct confidence index is determined by its confidence class. Thus, if a given CM assigns patterns to some class with the function $\mathrm{cm}(\mathbf{x}) = 1, ..., C$ the average classification error is defined as:

$$E_{cm} = \frac{1}{N_t} \sum_{n=1}^{N_t} |\mathrm{c}(\mathbf{x}^n) - \mathrm{cm}(\mathbf{x}^n)| \tag{5.4}$$

E_{cm} is called the *CLUES error* and quantifies the ability of a CM to correctly associate low standard deviation estimates with accurate predictions and high standard deviation with inaccurate ones. More specifically the CLUES error is the average misplacement, in terms of confidence regions, for a classified pattern. Ideally, E_{cm} should be zero.

The CLUES algorithm is summarised below.

1. **Define confidence regions.** The C confidence regions must take into account the true distribution of errors. The test set is also divided into C corresponding confidence classes.
2. **Compute class thresholds.** Thresholds s_i are computed using eq. (5.2).
3. **Compute confidence estimates.** Use the CM at hand to obtain an estimate of the prediction standard deviation for each test pattern.
4. **Classify test patterns.** Patterns are assigned to one of C confidence classes using the rule of eq. (5.3).
5. **Compute the CLUES error.** Use eq. (5.4) to compute the classification error.

The procedure for defining the confidence regions and other experimental details are described in the next section.

5.4 Defining the confidence regions

The comparison studies presented in this chapter use the same networks as the studies of chapter 4. As in chapter 4, the reported results are averaged over 500 committees formed by choosing networks at random from the pool of 300 trained networks. The committee size is again set to 20 networks for all methods.

In the previous chapter, the failure of PI CP to gauge local CM performance was demonstrated using artificial data sets of constant and input-dependent σ_ν^2. Here, we apply the CLUES technique to the same artificial data sets and to the curl data sets. The data sets are described in sections 4.4.2 and 4.4.3. The rest of this section presents the procedure for partitioning the input space and defining the confidence regions for the artificial and the curl data.

5.4.1 Defining confidence regions for the artificial data

For the purpose of these experiments, the input space is divided into five confidence regions which may be disjoint. In general, the number of confidence regions (and confidence levels) can be arbitrarily large as long as each region contains an adequate number of data. If a region contains few data, the CLUES error will not reflect possible misclassification of those accurately. The regions are chosen such that patterns in a particular region exhibit similar average error and training data PDF.

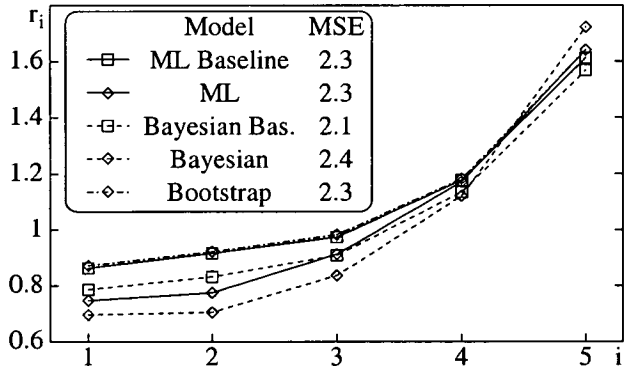


Figure 5.4: *The average prediction error per confidence region for artificial data set U using the five regression models.*

Partitions for the artificial data are defined by thresholding the direction of maximal variation of σ_ν^2 and PDF (*i.e.* their principal component). This direction is a linear combination of the

five inputs:

$$l(\mathbf{x}) = x_1 + x_2 - 2x_3 - 5x_4 + 2x_5 \tag{5.5}$$

Regions were defined by thresholding $l(\mathbf{x})$ using the following heuristic:

1. Divide the space into 10 regions by thresholding $l(\mathbf{x})$ so that each region contains equal numbers of test data.
2. Compute $<r(\mathbf{x})>$ for the data in the 10 regions and sort regions in order of increasing $<r(\mathbf{x})>$.
3. Group the consecutive regions in pairs to form 5 confidence regions.
4. Recalculate $<r(\mathbf{x})>$ for each confidence region to obtain r_i, $i = 1, ..., 5$. Region indices are sorted for increasing r_i.

For example, the average absolute prediction error r_i for each region is shown in fig. 5.4 for set U. The above algorithm leads to regions of approximately equal PDF and was chosen to automate region definition and to minimise manual intervention in the procedure.

Class thresholds are computed using eq. (5.2) using the prediction error estimates averaged over the regression models of the CMs under comparison.

5.4.2 Defining confidence regions for the curl data

As far as real data sets are concerned, the true distribution of errors is unknown. However, for the case of curl, problem-domain knowledge provides an indication of the actual form of σ_ν^2 through the method used to measure curl [22]. The curl measuring technique assumes that the curvature in the paper sample is small. Curl is then taken to be linearly proportional to the length of the shadow cast by the paper sheet when illuminated by a light source. This assumption is less accurate for large curl values. Large curl values are therefore likely to be corrupted by larger errors than are small values. A similar conclusion was reached during development of the initial curl model using constant σ_ν^2 [22], where it was found that the model is less accurate for large values of curl. In principle, this apparent increase in prediction error may also be partly attributable to the paucity of data with large curl values, *i.e.* due to small training data density rather than high data noise variance.

In this work, the CLUES technique is used to investigate the "measurement-error" hypothesis - that the noise variance in the curl data is input-dependent and depends on the curl value. Thus, no explicit assumption is adopted about the form that this relationship may take. Partitioning the data set based solely on the curl value is inaccurate because the targets are noisy. Such a partitioning corresponds to a partitioning of the input space which is unknown since the function that governs curl is unknown. However, the CLUES method does not require knowledge of the exact partitioning of the input space and partitions defined with respect to curl can be used as confidence regions for the curl problem. Therefore, the curl test sets (and input space) are partitioned with respect to the curl value as shown in tables 5.1 and 5.2.

Region	Curl	Test patterns (%)
1	0	28
2	5 - 10	23
3	15 - 25	24
4	30 - 45	17
5	50 - 85	8

Table 5.1: *Confidence Regions for the TR_8 test set.*

Region	Curl	Test patterns (%)
1	0	28
2	5 - 10	23
3	12 - 20	20
4	25 - 40	19
5	45 - 100	10

Table 5.2: *Confidence Regions for the TR_{11} test set.*

This particular choice of regions was made because:

- All patterns with the same curl value must be placed in the same region.
- The number of patterns per region must be approximately the same.
- The PDF of patterns in each region must be approximately the same.

A comparative study based on the CLUES technique will only be meaningful if the chosen input space partitioning reflects - with some accuracy - the true distribution of data error. In the rest of this section evidence that supports the particular choice of regions is presented.

Correspondence of confidence classes to confidence regions

It can be shown that a partition of the test set with respect to curl corresponds to a particular region of the input space by comparing the average Euclidean distance of pattern $\mathbf{x}^k$ in partition C_i from all other patterns in the same partition (average *local* distance $<d_i>$) to the average distance of $\mathbf{x}^k$ from all patterns that belong in all other partitions C_j, $j<>i$ (average *global* distance $<f_i>$). For a pattern $\mathbf{x}^k$ that belongs to C_i the local and global distances are given by:

$$d_i(\mathbf{x}^k) = \frac{1}{P_i - 1} \sum_{n}^{P_i} \|\mathbf{x}^k - \mathbf{x}^n\|^2 \tag{5.6}$$

$$f_i(\mathbf{x}^k) = \frac{1}{N_t - P_i} \sum_{m}^{N_t - P_i} \|\mathbf{x}^k - \mathbf{x}^m\|^2 \tag{5.7}$$

where $\mathbf{x}^n$ is a pattern in C_i other than $\mathbf{x}^k$, $\mathbf{x}^m$ is a pattern that does not belong to C_i and P_i is the total number of test patterns in partition C_i. Subsequently, d_i and f_i are computed for every point in C_i and the average over the points gives the average local and global distances ($<d_i>$ and $<f_i>$ respectively) for region i.

Region	Average local distance $<d_i>$	Average global distance $<f_i>$	Difference (%) $(<f_i>-<d_i>)/<f_i>$
1	13.1	13.9	5.7
2	12.1	11.4	-6.1
3	10.1	11.0	8.2
4	6.7	7.5	10.7
5	3.0	3.8	21.0

Table 5.3: *Average local and global distances for the TR_8 curl set partitions.*

Table 5.3 shows the results for the TR_8 test data. The global distance is larger than the local distance for all regions apart from region 2. This region contains data with curl ranging from 5 to 10mm. Interestingly, the two high curl regions (4 and 5) show markedly smaller local distances, as indicated by the relative difference (final column in the table).

Table 5.4 shows the results for the TR_{11} test data. The average local distance is smaller than the average global distance for all regions although by a smaller degree than before. Therefore, for both curl sets the result implies that most patterns of high curl do actually originate from a particular region of the input space.

Region	Average local distance $<d_i>$	Average global distance $<f_i>$	Difference (%) $(<f_i>-<d_i>)/<f_i>$
1	11.5	12.1	4.9
2	9.5	9.7	2.0
3	8.4	8.7	3.4
4	7.5	7.9	5.0
5	4.6	4.8	4.2

Table 5.4: *Average local and global distances for the TR_{11} curl set partitions.*

Average prediction error per class

Fig. 5.5 shows the r_i value per confidence region for set TR_8 using the five available regression models (*i.e.* the ML and Bayesian baseline and augmented models and the bootstrap model). It is obvious that there is an upward trend in the prediction error for all models. Moreover, the

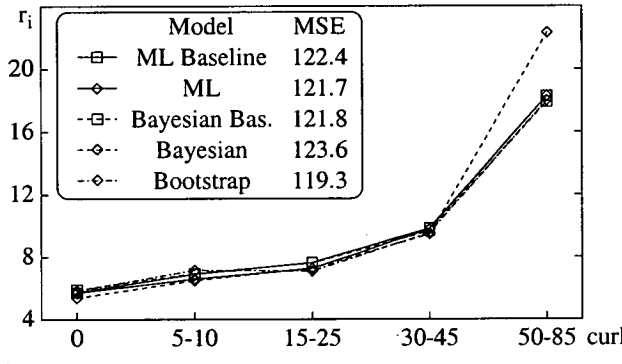


Figure 5.5: *The average prediction error for the confidence regions of the TR_8 curl set.*

r_i values are very similar for all five regression models, suggesting that the increasing trend is not a spurious effect. In principle, this trend may be due to increased data noise variance, low training data PDF, the presence of outliers or model misspecification. Here, it is assumed that the architecture of the curl model is optimised and that no gross outliers exist in the curl sets (this assumption agrees with the findings in [21]; however, note that work used a different data set).

Fig. 5.6 shows the r_i value per confidence region for set TR_{11} and all the available regression models. This time the average prediction error for region 2 is marginally lower than that of region 1. Thus, the regions are automatically sorted in order of increasing r_i so that the CLUES algorithm can be applied. Again the r_i values are very similar and r_i is higher in the regions of

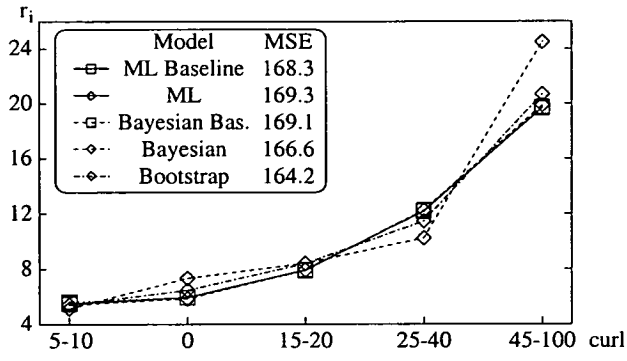


Figure 5.6: *The average prediction error for the confidence regions of the TR_{11} curl set.*

high curl.

Average PDF per class

Fig. 5.7 shows the normalised average PDF per confidence region for training sets TR_8 and

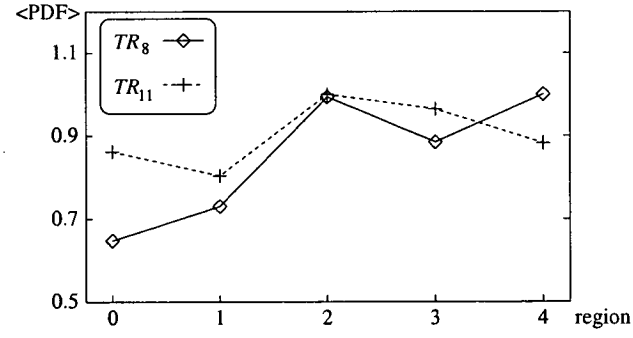


Figure 5.7: *The average PDF per confidence region for the curl training sets.*

TR_{11}. There is significant variation in the PDF between regions (particularly for set TR_8) but this partitioning is an acceptable choice for these small data sets. Although regions of high curl contain fewer points, the PDF is larger in the last three regions (regions of average and high curl) than in the first two (regions of low curl). This result implies that the increase in prediction error can not be wholly attributed to low training data density.

5.5 Results

Results of two comparison studies are presented in this section. First, CMs that treat data noise as constant are compared to CMs that consider input-dependent data noise. Then, the augmented network methods (ML and approximate Bayesian) are compared to the bootstrap technique. These comparisons were conducted in chapter 4 using the PI CP metric and the results were inconclusive as far as local performance is concerned. In this chapter they are repeated using the CLUES method as comparison metric.

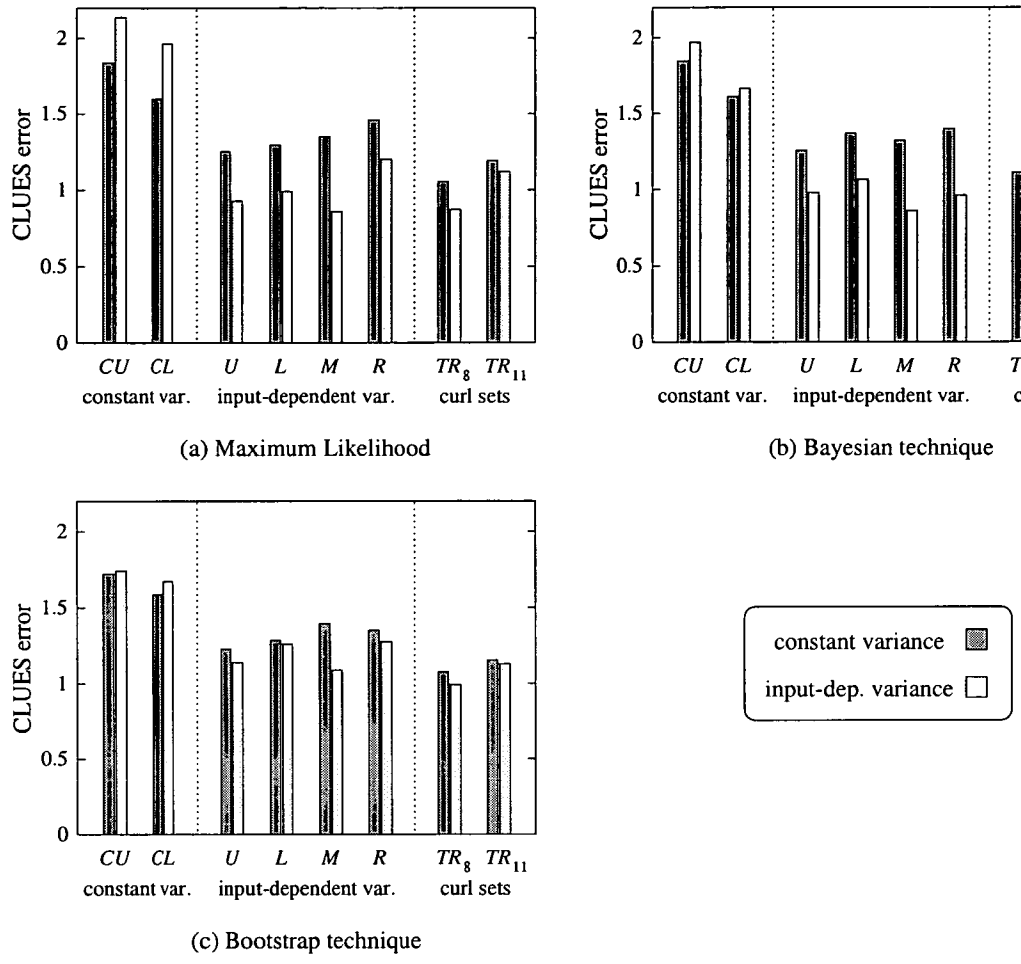
5.5.1 Comparison of CMs that treat noise variance as constant to CMs that treat noise variance as input-dependent

This study is a repeat of the study in section 4.5.6. The performance of CMs that treat noise variance as constant is compared to that of CMs that treat noise variance as input-dependent. In section 4.5.6 the PI CP was used as evaluation metric. Here, the CLUES method is used. Thus, the CMs are compared on the basis of their local performance. The main goal of this study is to investigate whether it is possible to gauge the input-dependency of the noise in the data by looking at the CLUES error variations.

The results are shown in fig. 5.8. The focus is on a comparison of the pairs of constant and input-dependent σ_ν^2 CMs rather than a global comparison across each graph. However, for reasons of uniformity, the CLUES error for the curl data is scaled so that the maximum possible error is the same for all data sets. The maximum CLUES error is different because the amount of data per class differs for artificial and curl data sets. For each CM pair, the thresholds have been computed by averaging the absolute prediction error values over the two regression models of the CMs under comparison.

In all cases the input-dependent σ_ν^2 ML and Bayesian CMs (fig. 5.8 (a) and (b)) perform better only for the input-dependent noise variance sets (sets *U*, *L*, *M* and *R*). Likewise, the ML and Bayesian constant σ_ν^2 CMs perform better only for sets of constant true noise variance (sets *CU*, and *CL*). As far as the curl set is concerned, both the ML and Bayesian CMs achieve significantly smaller CLUES error when the input-dependent σ_ν^2 assumption is used.

As far as the bootstrap method is concerned, the result is similar. The constant σ_ν^2 CMs achieve smaller CLUES error for the constant noise variance sets (*CU* and *CL*). The input-dependent σ_ν^2



(a) Maximum Likelihood

(b) Bayesian technique

(c) Bootstrap technique

Figure 5.8: *The CLUES error for comparison of CMs that consider constant noise variance to CMs that use input-dependent noise variance models using the (a) ML, (b) approximate Bayesian and (c) bootstrap techniques.*

CMs perform better for sets of input-dependent noise variance (sets U, L, M and R). Again, the input-dependent σ_ν^2 CM achieves smaller CLUES error for the curl set. However, the difference in performance is now smaller than in the case of the augmented network methods owing to the properties of the bootstrap algorithm. In the bootstrap method, data noise variance is estimated *after* regression modelling is completed, using the model residuals as targets for training a supplementary network. However, as explained in section 2.4.4, the bootstrap regression estimate is obtained under the assumption of constant data noise variance. Thus, both the constant and input-dependent noise variance CMs use the same regression model (and the same model residuals), leading to very similar results for the two assumptions.

The result is summarised in table 5.5 in which the average CLUES error is shown for sets of

constant, input-dependent and unknown true data noise variance.

Conclusions

The overall conclusion of this comparison study is that, unlike the PI CP metric (see chapter 4), the CLUES method can gauge local performance and can be used to infer the input-dependency of the noise in the data. All three methods exhibit better local confidence estimation performance for the curl data when the input-dependent σ_ν^2 method is used. This result supports the initial assertion about the dependency of the curl data noise variance on the curl value and therefore the input parameters.

actual variance	const. var. CM ■	input-dep. var. CM □
constant	1.7	2.1
input-dep	1.3	1.0
curl	1.1	1.0

(a) Maximum Likelihood

actual variance	const. var. CM ■	input-dep. var. CM □
constant	1.7	1.8
input-dep.	1.3	1.0
curl	1.3	0.9

(b) Bayesian technique

actual variance	const. var. CM ■	input-dep. var. CM □
constant	1.6	1.7
input-dep.	1.3	1.2
curl	1.1	1.0

(c) Bootstrap technique

Table 5.5: *The average CLUES error per category of sets for comparison of CMs that consider constant noise variance to CMs that use input-dependent noise variance models.*

5.5.2 Comparison of augmented network to conventional network architectures

In the previous study the input-dependency of the curl data noise was established. However, it is not clear if any benefits, in terms of local CM performance, arise when the regression model reflects the σ_ν^2 modelling decision. Therefore, in this study the confidence estimation performance of the augmented network methods is compared to that of the conventional network architectures. An analogous study in chapter 4 (section 4.5.5) showed that the approximate Bayesian approach is superior in terms of global performance. Here, the CLUES method is used to compare the local performance of the ML and Bayesian augmented network methods to that of the bootstrap method that uses a conventional network architecture. As before, the artificial data sets are used to investigate performance under controlled conditions.

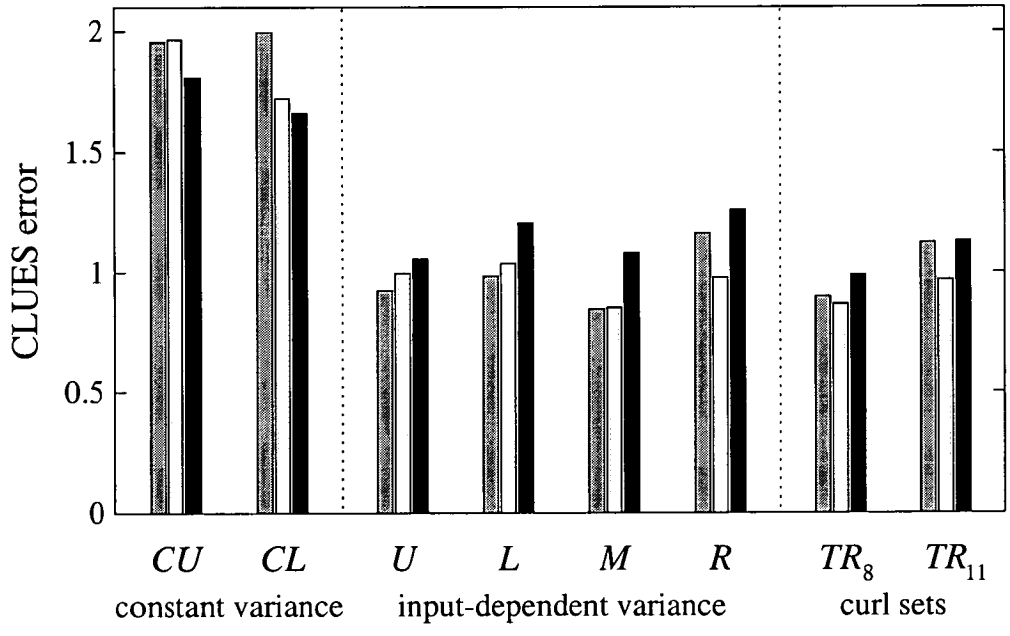


Figure 5.9: *The CLUES error for comparison of the ML, Bayesian and Bootstrap methods assuming input-dependent data noise variance.*

The result is shown in fig. 5.9. For each data set, the thresholds have been computed by averaging the absolute prediction error values over the three regression models of the CMs under comparison. For artificial data, the bootstrap method is superior to the augmented methods only when the actual noise in the data is constant. The augmented methods are superior for artificial sets of input-dependent σ_ν^2 as well as set TR_8. The Bayesian approach performs best for set TR_{11}. The result is summarised in table 5.6 in which the average CLUES error is shown for every category of data sets.

actual variance	ML ■	Bayes □	Bootstrap ■
constant	2.0	1.8	1.7
input-dep	1.0	1.0	1.2
curl	1.0	0.9	1.1

Table 5.6: *The average CLUES error for comparison of the ML, Bayesian and Bootstrap methods assuming input-dependent data noise variance.*

Although the difference in performance is small, it indicates that the bootstrap exhibits inferior local confidence estimation performance compared to the augmented network methods when the actual noise in the data is input-dependent. This failure can be attributed to the assumption implicit to the bootstrap approach, in which regression estimation is performed treating data noise variance as constant. The learning rate for the regression weights is therefore constant across input space and high noise regions are overemphasised. This inevitably affects the subsequent data noise variance estimation which uses the model residuals. In contrast, the augmented network methods, use an adaptive learning rate (which is inversely proportional to the σ_ν^2 estimate) for the regression weights. Network resources focus on low noise regions while high noise regions are de-emphasised [27]. This regime leads not only to an inherently different regression model but to a more realistic σ_ν^2 estimate as well.

Conclusions

The main conclusion of this comparison study is that taking into account the noise variance assumption when modelling the regression as well as the data noise variance leads to improved local CM performance.

5.6 Chapter summary and conclusions

This chapter presented a new approach to confidence measure comparison called classification of local uncertainty estimates (CLUES). CLUES compares CMs on the basis of the local quality of their standard deviation estimate. The method has been applied to a number of artificial problems of both constant and input-dependent real σ_ν^2 and to the real, curl problem.

The results for the artificial data (of known true data noise distribution) show that the CLUES error reflects the actual distribution of the noise in the data. In other words, CMs that treat noise variance as input-dependent yield smaller CLUES error when the actual noise variance is input-dependent. Similarly, CMs that treat noise variance as constant outperform (in terms of CLUES error) CMs that consider input-dependent noise variance when the actual noise variance is constant over input space. Other, commonly used, methods based upon the prediction interval coverage probability fail to indicate this because they cannot capture local CM performance.

As far as the curl sets are concerned, it was found that CMs that treat noise variance as input-dependent yield smaller CLUES error than CMs that consider constant noise variance. This result supports the initial assumption that the noise in the curl data is input-dependent and depends on the curl value.

The CLUES algorithm requires a choice of confidence regions (*i.e.* partitioning of the input space) that reflects - with a certain accuracy - the actual distribution of errors. In order to apply the CLUES method to the curl data, a partitioning based on the curl value (target) was used. This choice was indicated by the method used to measure curl and supported by the following findings:

- The high curl partitions correspond to particular regions of the input space which differ from the regions associated with low curl.
- The average prediction error per partition (for all available regression models) increases with curl.
- As the PDF was found to be larger in the high curl partitions, this trend could not be due to low data PDF solely.

These findings strongly support the hypothesis that data noise variance depends on curl.

CLUES was also used to compare the performance of CMs that use the augmented network architecture (ML and approximate Bayesian) to that of CMs that use the conventional network architecture (the bootstrap method which uses a supplementary network for input-dependent σ_ν^2 estimation). It was found that the augmented network methods outperform - in terms of local CM performance - the bootstrap method when the true noise variance is input-dependent. This result suggests that, when data noise is input-dependent, a more realistic regression and noise

variance model is obtained by estimating $y(\mathbf{x})$ and $\sigma_\nu^2(\mathbf{x})$ *simultaneously*, using an adaptive learning rate for the regression weights, rather than separately, using two independent networks.

In principle, CLUES may be used for classification problems as well as other (non-MLP) supervised learning models. The disadvantage is that it may only be applied if some *a priori* problem-domain knowledge or *intelligent guess* about the form of the distribution of data noise is available. As far as curl is concerned, the measurement method causes patterns of large curl to exhibit higher noise than patterns of low curl. In other similar applications partitioning with respect to one or more input variables may be appropriate.

Confidence estimation for the curl estimator

The findings of this chapter support the use of one of the augmented network methods for the curl estimator, since these methods exhibit better local CM performance than the bootstrap method. However, it has been found that the bootstrap method yields the best generalization error for the curl set. Although a much more principled training approach is necessary (*e.g.* trying different regularization parameters for each method) to establish the true potential of each method in terms of test error, the current test error trends can be accepted as indicative of each method's capabilities. This apparent contradiction highlights the trade-off between fitting all regions equally well and obtaining a locally accurate CM. In some applications, it may be preferable to sacrifice accuracy in high data noise regions in exchange for a better regression estimate and CM performance in relatively low data noise regions. For the curl problem, we are not only interested in predicting the actual curl value but also in predicting whether curl is going to be high or low [22], *i.e.* a classification problem. Pragmatically, predicting whether curl is going to be high or not is more crucial than predicting how high curl is likely to be. This is because high curl values (*e.g.* curl $\geq$ 30 or 40) are equally undesirable for the paper manufacturers [22]. Therefore, loss of accuracy in the (high noise) high curl region (in exchange of better local CM performance) is not a major concern.

For these reasons, an augmented network approach is more appropriate for the curl estimator. The approximate Bayesian approach yields an unbiased σ_ν^2 estimate (see chapter 2) and it was found to exhibit better local, as well as global, CM performance than the ML approach. Since training takes place off-line, we can use the approximate Bayesian approach although the training time required is about thirty times longer than that of the ML approach[2]. Finally, the

[2]The average training times for set TR_8 where about 30 and 1 hour respectively for a single network using a

findings presented here suggest that a reasonable modelling decision would be to incorporate the dependence of data noise variance on curl in the regression model. This can be done by connecting the $\sigma_\nu^2(\mathbf{x};\mathbf{u})$ output unit with the regression hidden units.

Overall conclusion

CLUES is a new practical method for comparing confidence estimation techniques on the basis of local performance. Using CLUES it was found that CMs that treat data noise variance as a function of the input achieve better local confidence estimation performance for the curl estimation problem, as well as for artificially generated data designed to highlight this. Moreover, it was shown that local CM performance improves when the assumption about the input-dependency of data noise variance is taken into account not only in modelling the noise variance, but in modelling the regression function as well.

Sun Ultra 10 workstation with a 332MHz processor.

Chapter 6
Summary and conclusions

6.1 Project summary

The overall aim of this project was to compare available confidence estimation methods (confidence measures) for neural networks and identify the most appropriate for use in a neural network industrial application, the prediction of paper curl. Three confidence measures were chosen as possible candidates: maximum likelihood (ML), the Bayesian approach with Gaussian approximation to the posterior (both backed by the delta method for model uncertainty estimation), and the bootstrap approach backed by a supplementary network for input-dependent data noise variance estimation. The reasons for this choice were:

- The chosen methods are commonly used and trusted by ANN researchers.
- At the time when this work was conducted more complex methods such as the exact Bayesian method using Monte Carlo integration and Gaussian processes were too computationally demanding and exceeded limitations set by the industry.
- The final methods should be easy to retrain by a non expert, preferably without manually set parameters.

Additionally, it was found that prediction and confidence estimation should be preceded by a novelty detection scheme that identifies and rejects novel inputs by thresholding the training data PDF. This is because techniques for model uncertainty variance estimation are unreliable in regions of very low training data density.

Having thus identified the candidate CM methods, attention was focused on techniques for CM performance evaluation and comparison. Two such methods were identified.

- In the first method, the actual model uncertainty variance is estimated by performing a Monte Carlo simulation over different realisations of the same artificial data. This approach was not followed here, because of its inherent disadvantages (see section 4.2.2).

- The second approach uses the properties of the confidence or prediction intervals (CIs or PIs) to evaluate CMs. The observed coverage probability (CP) of the intervals is estimated and compared with the theoretical value. A CM exhibits optimum performance if the observed coverage probability is consistently close to the theoretical value.

The CI CP and PI CP metrics were then used to evaluate and compare the candidate CMs. It was found that the CP can only gauge average CM performance over the entire input space. Consequently, a new approach to CM performance evaluation was developed to evaluate CMs on the basis of their local performance. The comparison studies were repeated using the new technique. The insight obtained during this work led to the formulation of a complete confidence estimation approach, appropriate for the curl problem.

CM comparison using the PI CP method: summary and conclusions

In order to assess CM performance a comparative study of the candidate CMs was performed using the PI CP method. This is the most commonly used approach to CM performance evaluation. Performance was assessed using the two first moments of the PI CP:

- The PI CP mean is a global measure of CM performance averaged over the entire input space.
- The PI CP standard deviation over the input space is a measure of the local quality of the confidence estimate.

Using sets of input-dependent as well as constant actual data noise variance, it was found that the approximate Bayesian approach supported by the approximate delta estimator yielded the most consistent mean CP estimate for both the constant and input-dependent noise variance assumptions. However, the CP standard deviation for this approach was larger than the one exhibited by the bootstrap technique.

Subsequently, a second comparison study was performed to compare CMs that treat data noise variance as constant to CMs that consider input-dependent data noise. It was found that the PI CP can not distinguish the performance of CMs that treat noise as constant from those that consider input-dependent noise. The reason is that the PI CP is an average over the input space metric and can not gauge CM performance locally. Therefore, it was concluded that the PI CP can not be used to gauge the input-dependency of the noise in the data.

The main conclusions of this work were:

1. The approximate Bayesian approach exhibits the most consistent observed mean PI CP, *i.e.* the best global performance.

2. The observed PI CP metric can only be used to assess the averaged quality of the confidence estimate and can provide no information about the local quality of the estimate.

The CLUES confidence measure evaluation method: summary and conclusions

Locally good confidence estimation performance is crucial in real applications. Therefore, a new CM comparison method, called classification of local uncertainty estimates (CLUES), was developed and used to assess local confidence estimation performance. Using artificial data, it was shown that the CLUES technique manages to capture the significant difference in performance between CMs that treat noise variance as constant and CMs that consider input-dependent noise variance.

The method was then applied to the curl data using the assumption that noise in the curl data depends on the curl value. This hypothesis was suggested by the method used to measure curl and supported by the data at hand. By dividing the curl data into subsets of similar curl, it was shown that the high curl subsets correspond to particular regions of the input space which differ from regions associated with low curl. Moreover, it was found that the average absolute prediction error per partition increased with curl for all available regression models. This trend could not be due to low data PDF solely as the PDF was actually larger in the high curl partitions. Finally, using the CLUES method it was found that CMs which adopt the input-dependent noise variance assumption exhibit locally better confidence estimation performance for the curl set.

The CLUES method was also used to compare the performance of CMs that use the augmented network architecture (ML and approximate Bayesian methods) with that of the bootstrap method which uses the conventional network architecture. In terms of local CM performance, the augmented network methods outperformed the bootstrap method when data noise variance was actually input-dependent. This result suggests that when the noise in the data is input-dependent (as it is in the case of curl data) more realistic regression and σ_ν^2 models are obtained by estimating $y(\mathbf{x})$ and $\sigma_\nu^2(\mathbf{x})$ *simultaneously* in an inter-linked manner, than separately using two independent networks.

The conclusions of this work were:

1. Using the augmented network architecture leads to better confidence estimation locally when the noise in the data is input-dependent.

2. The noise in the curl data set is input-dependent and increases with curl.

Confidence estimation for the curl task

The findings of chapters 4 and 5 suggest the use of an augmented network method for the curl estimator. The bootstrap technique was found to be more appropriate for data of constant noise variance, and it was shown that the noise in the curl data is input-dependent. However, the bootstrap method yielded the best generalisation error for the curl set. This contradiction can be viewed as a trade-off between fitting all regions equally well and obtaining a locally accurate CM. In some cases, it may be preferable to sacrifice accuracy in high data noise regions in exchange for a better regression estimate and CM performance in relatively low data noise regions. As far as the curl problem is concerned, loss of accuracy in the high curl region (curl $\geq$ 30 or 40) may not be crucial, since any high curl value is equally undesirable.

For these reasons, the use of an augmented network technique for the curl estimator is justified. The approximate Bayesian method is superior to the maximum likelihood approach because it achieves better global and local confidence estimation performance. ML methods have the advantage of requiring much shorter training times. Thus, they are more appropriate for applications in which small training times are crucial, *e.g.* when training is performed on-line. However, training the curl model can be performed off-line. Therefore, use of the approximate Bayesian approach is feasible and recommended for the curl estimator.

Finally, as any model uncertainty variance estimate can fail to identify novel inputs, confidence estimation should be preceded by a novelty detection stage which uses the training data PDF to identify and reject novel inputs.

6.2 Contribution: new knowledge in this thesis

A first, small, contribution is made in chapter 3 where it is shown that the uncertainty estimate obtained using the delta or bootstrap methods does not always increase rapidly outside the

region of adequate data density. Therefore, a novelty detection stage - based on evaluation of the training data probability density function - is recommended before a new input is presented to the network. This result is not clearly stated in the relevant literature.

Chapters 4 and 5 contain the main contributions to knowledge. A novel comparison study of CMs for MLPs is presented in chapter 4. This study uses the observed Prediction Interval Coverage Probability (PI CP) as evaluation criterion. It is the first study of this kind that considers the case of input-dependent data noise variance. The results show that Qazaz's approximate Bayesian approach [17] outperforms both ML [27] and Heskes's extended bootstrap approach [19]. On a different level, it is shown that neither the PI CP mean nor the PI CP standard deviation can gauge differences between confidence measures that treat data noise variance as constant or input-dependent. The PI CP can only be used as a *global* performance measure, over the entire input space.

Locally accurate confidence estimation is crucial therefore a new approach to confidence estimation evaluation is developed in chapter 5. The new approach measures *local* confidence estimation performance. No other similar CM evaluation technique has been reported in the relevant literature. The application of this approach to comparing the considered confidence estimation techniques leads to interesting conclusions. It is found that confidence measures that treat noise variance as input-dependent perform better when applied to the curl data. Therefore, the noise in the curl data must have input-dependent variance. This finding is also supported by the distribution of data density and model error. Moreover, the bootstrap technique, in which noise variance is implicitly considered constant during regression estimation, is found to be inferior (in terms of local performance) to methods that actively use the input-dependent noise variance assumption during regression estimation.

6.3 Conclusions and future work

The over-arching aim of this study was to examine the relationship between the characteristics of data, the network training approach and the optimal confidence estimation technique in the context of practical MLP applications. Thus, this thesis examined available methods for estimating confidence in neural network predictions as well as approaches to evaluating the performance of such methods. Before this project started these issues had received little systematic study. Therefore, little practical advice concerning confidence measures was available

for neural network system designers.

These issues were studied in conjunction with an industrial application, the prediction of paper curl. It was found that the noise in the curl data depends on the inputs and that both the regression and variance models should take this input-dependency into account. The approximate Bayesian approach backed with the approximate delta method is the most appropriate choice for the curl model implementation. Finally, the methodology used or developed during the course of this work may be applicable not only to curl modelling but also other similar real-world problems.

Further work

The results presented in this thesis were obtained using two curl data sets. It would be interesting to apply these methods to other real world problems. Other industrial or medical problems for which small data sets are available would be of particular interest. More specifically, it would be interesting to investigate the extent to which the CLUES algorithm can be used for other data sets.

As far as curl modelling is concerned, the findings of chapter 5 suggest that a reasonable modelling decision would be to directly incorporate the dependence of noise variance on curl in the model. This can be achieved by linking the regression hidden units with the variance output unit. Training can be performed in two stages. In the first stage an initial curl estimate is obtained while in the second one the whole network is fine-tuned. If this is a valid assumption, the above described model will outperform previous ones in terms of generalisation performance.

Finally, as technology progresses and more powerful computers become available at lower prices, more advanced confidence estimation techniques can be used in real system implementations. Therefore, in the following years, it may be necessary to extent this work to more advanced neural network models and confidence estimation techniques such as the exact Bayesian approach using the hybrid Monte Carlo technique, Gaussian Processes, and advanced bootstrap methods.

Appendix A
Training Regime

All networks were trained using weight decay regularization. The regularization parameter was constant and equal to 0.01 for all regression and variance networks. The only exception was the (baseline and augmented) Bayesian networks for set TR_{11} which were trained using the complete EF algorithm in which the regularization parameters values are adapted during training. This is because the augmented Bayesian networks for set TR_{11} form the latest version of the curl model.

The networks were trained for a fixed number of epochs, *i.e.* without checking for minimization of a validation error. Table A.1 gives the total number of epochs for each set.

Data Set	ML baseline	ML augmented	Bayesian baseline	Bayesian augmented	Bootstrap method
CU	25000	160000	25000	35000	40000
CL	25000	160000	22000	35000	40000
U	25000	160000	25000	35000	40000
L	25000	160000	25000	35000	40000
M	25000	120000	25000	35000	60000
R	35000	120000	35000	35000	80000
TR_8	10000	20000	10000	5000	10000
TR_{11}	10000	50000	10000	10000	10000

Table A.1: *Total number of epochs for each data set and training method.*

As explained in section 2.4.2 training for the augmented ML approach takes place in two stages. The table gives the total number of epochs. Each stage lasts for half of the total number of epochs.

Finally, when training the augmented Bayesian model for set TR_8, learning the variance is initially delayed for a number for epochs (300). This leads to improved regression learning.

Appendix B
The curl data

The curl data used in this thesis are the data used in previous curl modelling attempts [22, 134]. The data were collected in the Tullis Russell paper-making plant. Curl is measured after a reel of paper has been manufactured using a sample of paper taken from the end of the reel. The paper sheet is cut using a standard template. Subsequently, a shadow is cast (due to the curling paper) using a glancing angle light and curl is taken to be linearly proportional to the length of the shadow. This hypothesis is less accurate if the degree of curvature in the sample is large.

Data collection is very expensive in an industrial environment. Therefore, the available data are very sparse and there are many missing records. For this reason many of the available examples and input parameters had to be scrapped. Eventually, the following ten parameters were chosen for curl modelling:

1. percentage softwood
2. power used to process hardwood - refiner 1
3. power used to process hardwood - refiner 2
4. percentage broke
5. caliper (paper thickness) at the paper machine
6. ash content
7. moisture content at the paper machine
8. porosity measure
9. difference between surface moisture levels after topcoat application
10. paper grade

All these parameters are automatically measured except "porosity measure", which is measured by a human operator, and "paper grade", whose value is, naturally, fixed. These parameters were chosen for reasons of availability and through experimentation.

The database contains paper samples of three grades. The "paper grade" symbolic input is encoded using two input units as explained in [22]. Therefore, the final number of inputs is 11 (instead of 12 units that would be necessary if conventional encoding was used). The data are then pre-processed using the Karhunen-Loève transformation [35] that extracts the principal components of the data. For more details about data collection and pre-processing see [22].

The database contains 672 examples. At first, the curl model was developed using the largest eight principal components as inputs for the neural network [22]. That version used only 554 of the available patterns as it worked only for a particular grade of paper. In this thesis, that data set is denoted TR_8. The current version of the model uses all the available data and all eleven principal components. The complete data set is denoted TR_{11}.

Appendix C
Publications

1. G. Papadopoulos, P.J. Edwards and A.F. Murray, "Confidence Estimation Methods for Neural Networks: A Practical Comparison", in ESANN'2000, European Symposium on Artificial Neural Networks, April 26-27-28, 2000 (M. Verleysen, ed.) (Bruges, Belgium), pp. 75-80, 2000

2. G. Papadopoulos, P.J. Edwards and A.F. Murray, "A Practical Comparison of Confidence Estimation Methods for Neural Networks", Proceedings of the Fifth International Conference on Cognitive and Neural Systems, (Boston, MA, USA), 2000

3. G. Papadopoulos, P.J. Edwards and A.F. Murray, "Confidence Estimation Methods for Neural Networks: A Practical Comparison", submitted to IEEE Transactions in Neural Networks

4. G. Papadopoulos, P.J. Edwards and A.F. Murray, "CLUES: A Region-Dependent Approach to the Comparison of Neural Network Prediction Uncertainty Estimates", submitted to IEEE Transactions in Neural Networks

References

[1] W. G. Baxt and H. White, "Bootstrapping confidence intervals for clinical input variable effects in a network trained to identify the presence of acute myocardial infraction," *Neural Computation*, vol. 7, no. 3, pp. 624–638, 1995.

[2] R. Lippmann, L. Kukolich, and D. Shahian, "Predicting the risk of complications in coronary artery bypass operations using neural networks," in *Advances in Neural Information Processing Systems 7, MIT Press, Cambridge, MA, USA, 1995, xxi+1143 pp.*, vol. 7, pp. 1055–1062, MIT Press, 1995.

[3] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady, "Novelty detection for the identification of masses in mammograms," in *Proceedings of the Fourth International IEE Conference on Artificial Neural Networks (Cambridge, 1995)*, vol. 409 of *IEE Conference Publication*, pp. 442–447, IEE, 1995.

[4] E. A. Wan, "Time series prediction by using a connectionist network with internal delay lines," in *Time Series prediction: Forecasting the Future and Understanding the Past* (A. S. Weigend and N. A. Gershenfeld, eds.), vol. XV of *Santa Fe Institute Studies in the Science of Complexity*, (Reading, MA), pp. 195–217, Addison-Wesley, 1994.

[5] M. C. Mozer, "Neural net architectures for temporal sequence processing," in *Time Series prediction: Forecasting the Future and Understanding the Past* (A. S. Weigend and N. A. Gershenfeld, eds.), vol. XV of *Santa Fe Institute Studies in the Science of Complexity*, (Reading, MA), pp. 243–264, Addison-Wesley, 1994.

[6] M. Cottrell, B. Girard, and P. Rousset, "Long term forecasting by combining Kohonen algorithm and standard prevision," in *Proc. ICANN'97, 7th International Conference on Artificial Neural Networks*, vol. 1327 of *Lecture Notes in Computer Science*, pp. 993–998, Berlin: Springer, 1997.

[7] D. Ranaweera, N. Hubele, and A. Papalexopoulos, "Application of radial basis function neural network model for short-term load forecasting," *IEE Proc.-Gener. Transm. Distrib.*, vol. 142, no. 1, pp. 45–50, 1995.

[8] P. Lajbcygier and J. Connor, "Improved option pricing using bootstrap methods," in *1997 IEEE International Conference on Neural Networks Proceedings*, vol. 4, pp. 2193–2197, IEEE, 1997.

[9] C. M. Bishop, "Neural network validation: an illustration from the monitoring of multiphase flows," in *In Proceedings IEE Third International Conference on Artificial Neural Networks, Brighton, UK*, pp. 41–45, 1993.

[10] T. Petsche, A. Marcantonio, C. Darken, S. Hanson, G. Kuhn, and I. Santoso, "A neural network autoassociator for induction motor failure prediction," in *Advances in Neural Information Processing Systems 8, Proceedings of the 1995 Conference. MIT Press, Cambridge, MA, USA, 1996, xix+1098 pp.*, pp. 924–930, MIT Press, 1996.

[11] S. Cho, Y. Cho, and S. Yoon, "Reliable roll force prediction in cold mill using multiple neural networks," *IEEE Transactions on Neural Networks*, vol. 8, pp. 874–882, July 1997.

[12] N. Pican, F. Alexandre, and P. Bresson, "Artificial neural networks for the presetting of a steel temper mill," *IEEE Expert*, vol. 11, no. 1, pp. 22–27, 1996.

[13] J. Zhang, E. Martin, A. Morris, and C. Kiparissides, "Inferential estimation of polymer quality using stacked neural networks," *Computers and Chemical Engineering, suppl. issue*, vol. 21, no. 1, pp. 1025–1030, 1997.

[14] J. N. Fidalgo, M. A. Matos, and M. T. P. de Leao, "Assessing error bars in distribution load curve estimation," in *Artificial Neural Networks—ICANN '97. 7th International Conference Proceedings* (W. Gerstner, A. Germond, M. Hasler, and J. D. Nicoud, eds.), pp. 1017–22, Berlin, Germany: Springer-Verlag, 1997.

[15] J. R. Donaldson and R. B. Schnabel, "Computational experience with confidence regions and confidence intervals for nonlinear least squares," *Technometrics*, vol. 29, no. 1, pp. 67–82, 1987.

[16] J. T. G. Hwang and A. A. Ding, "Prediction intervals for artificial neural networks," *Journal of the American Statistical Association*, vol. 92, pp. 748–757, June 1997.

[17] C. Qazaz, *Bayesian Error Bars for Regression.* PhD thesis, Aston University, 1996.

[18] C. M. Bishop, "Novelty detection and neural network validation," in *IEE Proceedings in Vision, Image and Signal Processing*, vol. 141, pp. 217–222, 1994.

[19] T. Heskes, "Practical confidence and prediction intervals," in *Advances in Neural Information Processing Systems* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), vol. 9, pp. 176–182, The MIT Press, 1997.

[20] R. D. D. Veaux, J. Schumi, J. Schweinsberg, and L. H. Ungar, "Prediction intervals for neural networks via nonlinear regression," *Technometrics*, vol. 40, no. 4, pp. 273–282, 1998.

[21] A. Myles, *Methods for addressing some practical issues in MLP regression and their application to modelling curl in papermaking.* PhD thesis, University of Edinburgh, 1997.

[22] P. J. Edwards, A. F. Murray, G. Papadopoulos, A. R. Wallace, J. Barnard, and G. Smith, "The application of neural networks to the papermaking industry," *IEEE Transactions on neural networks*, vol. 10, pp. 1456–1464, November 1999.

[23] G. Chryssolouris, M. Lee, and A. Ramsey, "Confidence interval prediction for neural network models," *IEEE Transactions on Neural Networks*, vol. 7, pp. 229–232, Jan. 1996.

[24] R. Shao, E. Martin, J. Zhang, and A. Morris, "Confidence bounds for neural network representations," *Computers and Chemical Engineering, suppl. issue*, vol. 21, no. 1, pp. 1173–1178, 1997.

[25] I. Rivals and L. Personnaz, "Construction of confidence intervals for neural networks based on least squares estimation," *Neural Networks*, vol. 1, no. 13, pp. 463–484, 2000.

[26] D. A. Nix and A. S. Weigend, "Estimating the mean and variance of the target probability distribution," in *Proceedings of the IJCNN'94*, pp. 55–60, IEEE, 1994.

[27] D. A. Nix and A. S. Weigend, "Learning local error bars for nonlinear regression," in *Advances in Neural Information Processing Systems* (G. Tesauro, D. Touretzky, and T. Leen, eds.), vol. 7, pp. 489–496, The MIT Press, 1995.

[28] P. M. Williams, "Using neural networks to model conditional multivariate densities," *Neural Computation*, vol. 8, no. 4, pp. 843–854, 1996.

[29] W. Buntine and A. S. Weigend, "Bayesian back-propagation.," *Complex Systems*, vol. 5, no. 6, pp. 603–643, 1991.

[30] D. MacKay, "A practical Bayesian framework for backprop networks," *Neural Computation*, vol. 4, no. 3, pp. 448–472, 1992.

[31] R. M. Neal, "Bayesian training of backpropagation networks by the hybrid Monte Carlo method," Tech. Rep. 92/01, University of Toronto, Apr. 1992.

[32] P. Williams, "Bayesian regularization and pruning using a Laplace prior," *Neural Computation*, vol. 7, no. 1, pp. 117–143, 1995.

[33] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. Chapman and Hall, London, UK, 1993.

[34] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 1995.

[35] I. Jollife, *Principal Components Analysis*. New York: Springer-Verlag, 1986.

[36] M. C. Mozer and P. Smolensky, "Skeletonization: a technique for trimming the fat for a network via relevance assessment," in *Advances in Neural Information Processing Systems* (D. S. Touretzky, ed.), vol. 1, (San Mateo), pp. 107–115, CA: Morgan Kaufmann, 1989.

[37] Y. L. Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, vol. 2, pp. 598–605, 1990.

[38] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: optimal brain surgeon," in *Advances in Neural Information Processing Systems* (S. J. Hanson, J. D. Cowan, and C. L. Giles, eds.), vol. 5, pp. 164–171, 1993.

[39] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems* (D. S. Touretzky, ed.), vol. 2, (San Mateo), pp. 524–532, CA: Morgan Kaufmann, 1990.

[40] M. G. Bello, "Enhanced training algorithms and integrated training/architecture selection for multilayer perceptron networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 864–875, 1992. BisBook.

[41] D. MacKay, "Bayesian interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415–447, 1992.

[42] L. K. Jones, "Constructive approximations for neural networks by sigmoidal functions," *Proceedings of the IEEE*, vol. 10, no. 78, pp. 1586–1589, 1990.

[43] E. K. Blum and L. K. Li, "Approximation theory and feedforward networks," *Neural Networks*, vol. 4, no. 4, pp. 511–515, 1991.

[44] G. A. F. Seber, *Linear Regression Analysis*, ch. 3. New York: Wiley, 1977.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representation by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (D. E. Rumelhart and J. L. McClelland, eds.), vol. 1, pp. 318–362, Cambridge, MA: MIT Press, 1986.

[46] D. Plaut, S. Nowlan, and G. Hinton, "Experiments on learning by back propagation," Technical Report CMU-CS-86-126, Dep. of Computer Science, Carnegie Mellon University, 1986.

[47] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the back-propagation algorithm," *Biological Cybernetics*, vol. 59, pp. 257–263, 1988.

[48] R. Battiti, "Accelerated backpropagation learning: two optimization methods," *Complex Systems*, vol. 3, pp. 331–342, 1989.

[49] S. G. Nash, "Truncated Newton methods," Tech. Rep. STAN-CS-82-906, Dept. of Computer Science, Stanford Univ., 1982.

[50] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C, 2nd. edition*. Cambridge University Press, 1992.

[51] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[52] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *Computer Journal (British Computer Society)*, vol. 7, pp. 149–154, 1964.

[53] R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization," *Computer Journal (British Computer Society)*, vol. 6, pp. 163–168, 1963.

[54] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, p. 119. London: Academic Press, 1981.

[55] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly Journal of Applied Mathematics*, vol. II, no. 2, pp. 164–168, 1944.

[56] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society of Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[57] D. Hush and J. Salas, "Improving the learning rate of back-propagation with the gradient re-use algorithm," in *IEEE International Conference on Neural Networks*, vol. 1, (San Diego), pp. 441–447, CA: IEEE, 1988.

[58] R. Brent, *Algorithms for Minimization without Derivatives*. Englewood Cliffs, NJ: Prentice-Hall, 1973.

[59] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society, B*, vol. 36, no. 1, pp. 111–147, 1974.

[60] M. Stone, "Cross-validation: a review," *Math. Operationsforsch. Statist. Ser. Statistics*, vol. 9, no. 1, pp. 127–139, 1978.

[61] G. Wahba and S. Wold, "A completely automatic french curve: fitting spline functions by cross-validation," *Communications in Statistics, Series A*, vol. 4, no. 1, pp. 1–17, 1975.

[62] G. Hinton, "Learning translation invariant recognition in massively parallel networks," in *Proceedings, PARLE Conference on Parallel Architectures and Language Europe*, pp. 1–13, Berlin: Springer-Verlag, 1987.

[63] V. Tresp, S. Ahmad, and R. Neuneier, "Training neural networks with deficient data," in *Advances in Neural Information Processing Systems* (J. D. Cowan, G. Tesauro, and J. Alspector, eds.), vol. 6, pp. 128–135, Morgan Kaufmann Publishers, Inc., 1994.

[64] D. K. Ranaweera, G. G. Karady, and R. G. Farmer, "Effect of probabilistic inputs on neural-network based electric load forecasting," *IEEE Transactions on Neural Networks*, vol. 7, pp. 1528–1532, Nov. 1996.

[65] R. Hogg and J. Ledolter, *Applied Statistics for Engineers and Physical Scientists*. Maxwell Macmillan, 1992.

[66] G. B. Wetherill, *Regression Analysis with Applications*, ch. 9. London: Chapman & Hall, 1986.

[67] G. A. F. Seber and C. J. Wild, *Nonlinear Regression*, ch. 5. New York: Wiley, 1989.

[68] D. Lowe and K. Zapart, "Point-wise confidence interval estimation by neural networks: A comparative study based on automotive engine calibration.," *Neural Computing and Applications*, vol. 8, no. 1, pp. 77–85, 1999.

[69] R. Bellman, *Adaptive Control Processes: A Guided Tour*. New Jersey: Princeton University Press, 1961.

[70] G. B. Wetherill, *Regression Analysis with Applications*, ch. 6. London: Chapman & Hall, 1986.

[71] R. M. Neal, *Bayesian Learning for Neural Networks*. No. 118 in Lecture Notes in Statistics, New York: Springer, 1996.

[72] G. A. F. Seber, *Linear Regression Analysis*, ch. 5. New York: Wiley, 1977.

[73] R. Tibshirani, "A comparison of some error estimates for neural network models," *Neural Computation*, vol. 8, no. 1, pp. 152–163, 1996.

[74] J. C. Nash and M. Walker-Smith, *Nonlinear Parameter Estimation*, ch. 11. New York: Marcel Dekker, Inc., 1987.

[75] C. Bishop, "Exact calculation of the Hessian matrix for the multilayer perceptron," *Neural Computation*, vol. 4, no. 4, pp. 494–501, 1992.

[76] W. L. Buntine and A. S. Weigend, "Computing second derivatives in feed-forward networks: A review," *IEEE Transactions on Neural Networks*, vol. 5, pp. 480–488, May 1994.

[77] J. R. Westlake, *A Handbook of Numerical Matrix Inversion and Solution of Linear Equations*. New York: Wiley, 1968.

[78] G. Forsythe, M. Malcolm, and C. Moler, *Computer Methods for Mathematical Computations*, ch. 9. Englewood Cliffs, NJ: Prentice-Hall, 1977.

[79] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Baltimore: University Press, 2nd. ed., 1989.

[80] J. Wilkinson and C. Reinsch, *Linear Algebra, vol II of Handbook for Automatic Computation*, ch. I.10. New York: Springer-Verlag, 1971.

[81] J. M. Chambers, *Computational Methods for Data Analysis*. New York: Wiley, 1977.

[82] J. Stoer and L. Bulirsch, *Introduction to Numerical Analysis*, ch. 6.7. New York: Springer-Verlag, 1980.

[83] Y. Bard, *Nonlinear Parameter Estimation*. New York/London: Academic Press, 1974.

[84] G. A. F. Seber, *Applied Regression Analysis and Other Multivariable Methods*, ch. 21. Boston: PWS-KENT, 1988.

[85] J. C. Nash and M. Walker-Smith, *Nonlinear Parameter Estimation*, ch. 2. New York: Marcel Dekker, Inc., 1987.

[86] C. Satchwell, "Finding error bars (the easy way)," in *Neural Computing Applications Forum, Edition 5*, 1994.

[87] J. O. Rawlings, *Applied Regression Analysis*, ch. 9. Wadsworth and Brooks/Cole, 1988.

[88] C. M. Bishop, "Mixture density networks," Tech. Rep. NCRG/94/004, Aston University, 1994.

[89] D. MacKay, "Evidence framework applied to classification networks," *Neural Computation*, vol. 4, no. 5, pp. 720–736, 1992.

[90] H. H. Thodberg, "A review of Bayesian neural networks with an application to near infrared spectroscopy," *IEEE Transactions on Neural Networks*, vol. 7, pp. 56–72, Jan. 1996.

[91] C. M. Bishop and C. S. Qazaz, "Regression with input-dependent noise: A bayesian treatment," in *Advances in Neural Information Processing Systems* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), vol. 9, pp. 347–353, The MIT Press, 1997.

[92] R. M. Neal, "Bayesian learning via stochastic dynamics," in *Advances in Neural Information Processing Systems 5. Proceedings of the 1992 Conference* (S. J. Hanson, J. D. Cowan, and C. L. Giles, eds.), (San Mateo, CA), pp. 475–482, Morgan Kaufmann, 1993.

[93] D. MacKay, "Probabilistic networks: new models and new methods," in *Proceedings of ICANN'95*, 1995.

[94] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

[95] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, pp. 721–741, 1984.

[96] H. C. Andersen, "Molecular dynamics simulations at constant pressure and/or temperature," *Journal of Chemical Physics*, vol. 72, pp. 2384–2393, 1980.

[97] S. Duane, A. Kennedy, B. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics Letters*, vol. 195, no. 2, pp. 216–222, 1987.

[98] D. MacKay, "Hyperparameters: optimize or integrate out?," in *Maximum Entropy and Bayesian Methods* (G. Heidbreder, ed.), (Santa Barbara 1993), pp. 43–59, Dordrecht: Kluwer, 1994.

[99] D. Wolpert and C. Strauss, "What bayes has to say about the evidence procedure," in *Maximum Entropy and Bayesian Methods* (G. Heidbreder, ed.), (Santa Barbara 1993), pp. 61–78, Dordrecht: Kluwer, 1994.

[100] C. Rodriguez, "Objective bayesianism and geometry," in *Maximum Entropy and Bayesian Methods* (P. Fougére, ed.), (Norwell, MA), pp. 61–78, Kluwer, 1990.

[101] P. Williams, "Improved generalization and network pruning using adaptive laplace regularization," in *Proceedings of the Third International Conference on Artificial Neural Networks*, pp. 76–80, IEE, 1994.

[102] G. Hinton and D. van Camp, "Keeping neural networks simple by minimizing the description length of the weights," in *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pp. 5–13, 1993.

[103] K. Kim and E. B. Bartlett, "Error estimation by series association for neural network systems," *Neural Computation*, vol. 7, no. 4, pp. 799–808, 1995.

[104] B. D. Ripley, *Statistical Inference for Spatial Processes*. Cambridge: Cambridge University Press, 1988.

[105] C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems* (D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, eds.), vol. 8, pp. 514–520, The MIT Press, 1996.

[106] D. MacKay, "Gaussian processes," in *NIPS 97 Tutorials* (L. F. Niklasson and M. B. Boden, eds.), pp. 165–178, MIT Press, 1997.

[107] D. Barber and C. K. I. Williams, "Gaussian processes for Bayesian classification via hybrid Monte Carlo," in *Advances in Neural Information Processing Systems* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), vol. 9, p. 340, The MIT Press, 1997.

[108] R. M. Neal, "Monte Carlo implementation of gaussian process models for Bayesian regression and classification," Tech. Rep. 97/02, University of Toronto, Jan. 1997.

[109] C. K. I. Williams, "Computing with infinite networks," in *Advances in Neural Information Processing Systems* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), vol. 9, pp. 295–301, The MIT Press, 1997.

[110] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," in *Artificial Neural Networks for Speech and Vision* (R. J. Mammone, ed.), (London), pp. 126–142, Chapman & Hall, 1993.

[111] M. P. Perrone, "Putting it all together: Methods for combining neural networks," in *Advances in Neural Information Processing Systems* (J. D. Cowan, G. Tesauro, and J. Alspector, eds.), vol. 6, pp. 1188–1189, Morgan Kaufmann Publishers, Inc., 1994.

[112] B. Parmanto, P. W. Munro, and H. R. Doyle, "Improving committee diagnosis with resampling techniques," in *Advances in Neural Information Processing Systems* (D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, eds.), vol. 8, pp. 882–888, The MIT Press, 1996.

[113] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[114] L. Breiman, "Stacked regressions," Tech. Rep. TR 367, Dept. of Statistics, UC. Berkeley, 1992.

[115] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[116] T. Heskes, "Balancing between bagging and bumping," in *Advances in Neural Information Processing Systems* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), vol. 9, p. 466, The MIT Press, 1997.

[117] P. Edwards and A. Murray, "Committee formation for reliable and accurate neural prediction in industry," in *ESANN'2000, European Symposium on Artificial Neural Networks, April 26-27-28, 2000* (M. Verleysen, ed.), (Bruges, Belgium), pp. 141–146, 2000.

[118] S. Hashem, *Optimal Linear Combinations of Neural Networks*. PhD thesis, Purdue University, 1994.

[119] G. A. F. Seber, *Applied Regression Analysis and Other Multivariable Methods*, ch. 3. Boston: PWS-KENT, 1988.

[120] L. Parra, G. Deco, and S. Miesbach, "Statistical independence and novelty detection with information preserving nonlinear maps," *Neural Computation*, vol. 8, no. 2, pp. 260–269, 1996.

[121] M. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChe Journal*, vol. 37, no. 2, pp. 233–243, 1991.

[122] C. Williams, C. Qazaz, C. Bishop, and H. Zhu, "On the relationship between Bayesian error bars and the input data density," in *Proceedings of the Fourth International IEE Conference on Artificial Neural Networks (Cambridge, 1995)*, vol. 409 of *IEE Conference Publication*, pp. 160–165, IEE, 1995.

[123] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall, 1986.

[124] H. G. C. Travèn, "A neural network approach to statistical pattern classification by "semiparametric" estimation of probability density functions," *IEEE Transactions on Neural Networks*, vol. 2, no. 3, pp. 366–377, 1991.

[125] S. Roberts and L. Tarassenko, "A probabilistic resource allocating network for novelty detection," *Neural Computation*, vol. 6, no. 2, pp. 270–284, 1994.

[126] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *Annals of Mathematical Statistics*, vol. 27, pp. 832–837, 1956.

[127] E. Parzen, "On the estimation of a probability density function and mode," *Annals of Mathematical Statistics*, vol. 33, pp. 1065–1076, 1962.

[128] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, vol. 55 of *Applied Mathematics Series*. New York: Dover Publications, 1968.

[129] R. B. D'Agostino and M. A. Stephens, *Goodness-of-fit techniques*. New York: Marcel Dekker, Inc., 1986.

[130] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992.

[131] J. Freidman, "Multivariate adaptive regression splines," *Annals of Statistics*, vol. 19, pp. 1–141, 1991.

[132] J. A. Leonard, M. A. Kramer, and L. H. Ungar, "A neural network architecture that computes its own reliability," *Computers and Chemical Engineering*, vol. 16, pp. 819–835, September 1992.

[133] J. A. Leonard, M. A. Kramer, and L. H. Ungar, "Using radial basis functions to approximate a function and its error bounds," *IEEE Transactions on Neural Networks*, vol. 3, pp. 624–627, July 1992.

[134] P. Edwards, A. Murray, G. Papadopoulos, A. Wallace, and J. Barnard, "The application of neural networks to the paper-making industry," in *ESANN'99, European Symposium on Artificial Neural Networks, April 21-22-23, 1999* (M. Verleysen, ed.), (Bruges, Belgium), pp. 69–74, 1999.